Cobbler Documentation

Release 2.8.5

Jörgen Maas

Mar 23, 2022

Contents

1 About **3**
- 1.1 Release Notes . . . 3
- 1.2 Distribution Support . . . 5
- 1.3 Distribution Notes . . . 7
- 1.4 How We Model Things . . . 13

2 Installation **15**
- 2.1 Prerequisites . . . 15
- 2.2 Installing from packages . . . 16
- 2.3 Installing from Source . . . 18
- 2.4 Configuration Files . . . 20
- 2.5 Relocating your installation . . . 21

3 General **23**
- 3.1 Cobbler Primitives . . . 23
- 3.2 Cobbler Direct Commands . . . 54
- 3.3 Cobbler Settings . . . 69
- 3.4 Managing Services with Cobbler . . . 90
- 3.5 Kickstart Templating . . . 94
- 3.6 Snippets . . . 102
- 3.7 Package Management and Mirroring . . . 112
- 3.8 File System Information . . . 114

4 Advanced **119**
- 4.1 Advanced Networking . . . 119
- 4.2 SELinux . . . 122
- 4.3 Configuration Management . . . 123
- 4.4 Extending cobbler . . . 131
- 4.5 Power Management . . . 135
- 4.6 Alternative template formats . . . 138
- 4.7 Multi-Homes cobbler servers . . . 138
- 4.8 Auto registration . . . 139

4.9 Batch editing . . . 139
4.10 Moving to a new server . . . 140
4.11 PXE boot-menu passwords . . . 140
4.12 Alternative storage backends . . . 142
4.13 Using gPXE . . . 146
4.14 Data revision control . . . 148
4.15 PXE behaviour and tailoring . . . 148
4.16 Kickstart Tracking . . . 150
4.17 Boot CD . . . 150

5 Web Interface **151**
5.1 Security Overview . . . 151
5.2 Web Authentication . . . 152
5.3 Web Authorization . . . 159
5.4 Locking down cobbler . . . 161
5.5 Basic Setup . . . 162
5.6 Basic setup (2.2.x and higher) . . . 163

6 Frequently Asked Trouble Shooting Questions **165**
6.1 General . . . 165
6.2 Debugging Cobbler Web . . . 168
6.3 Hints and tips: Redhat . . . 168
6.4 Frequently Asked Virtualization Trouble Shooting Questions . . . 169

7 Appendix **171**
7.1 S390 Support . . . 171
7.2 Power PC support . . . 174
7.3 Tips for RHN . . . 178
7.4 Memtest . . . 179
7.5 Anaconda Monitoring . . . 180
7.6 System Retirement . . . 182
7.7 Booting Live CD's . . . 184
7.8 Clonezilla Integration . . . 185

8 Indices and tables **187**

cobbler is a provisioning (installation) and update server. It supports deployments via PXE (network booting), virtualization (Xen, QEMU/KVM, or VMware), and re-installs of existing Linux systems. The latter two features are enabled by usage of 'koan' on the remote system. Update server features include yum mirroring and integration of those mirrors with automated installation files. Cobbler has a command line interface, Web UI, and extensive Python and XMLRPC APIs for integration with external scripts and applications.

Here you should find a comprehensive overview about the usage of cobbler.

CHAPTER 1

About

1.1 Release Notes

1.1.1 2.8.0

Deprecation warnings

The following list of features have been deprecated and will *not* be available in Cobbler 3.0. Users depending on one of these features should stick to the 2.8.x (LTS) series. If you'd like to maintain one of these features in 3.x and beyond, then please reach out to the developers through GitHub or the mailinglist.

- MySQL backend
- CouchDB backend
- MongoDB backend
- Monit support
- Platforms: s390/s390x and ia64
- System field names: subnet, bonding_master, bonding
- Func integration
- Koan LiveCD
- Koan LDAP configuration
- redhat_management (Spacewalk/Satellite)
- Remote kickstart files/templates (other than on the Cobbler server)
- The concurrent use of `parent` and `distro` on subprofiles

Feature improvements

- Signature updates: Fedora 24/25, Ubuntu 16.10, Virtuozzo 7
- Integrated `pyflakes` into the build system and fixed all reported issues
- Integrated Travis CI into the build system (`make qa`)
- Allow https method in repo management (#1587)
- Add support for the `ppc64le` architecture
- Backport gpxe mac search argument
- Added support for fixed DHCP IPs when using vlan over bond
- Add support for Django 1.7.x and 1.8.x
- Add action name to cobbler action `--help` output

Bugfixes

- Added HOSTS_ALLOW acl in settings.py (CVE-2016-9014)
- Profile template logic seperated for grub and pxelinux formats
- Refer to system_name in grubsystem.template
- Add netmask and dhcp_tag to slave interfaces in ISC DHCP
- Koan now works with CentOS version numbers
- Fixes to pxesystem_esxi.template
- Move `get-loaders` to https transport
- Add default/timeout to grubsystem.template
- Anamon now actually waits on files that you specify with `--watchfiles`
- Do not set interface["filename"] to /pxelinux.0 in manage_isc.py (#1565)
- Allow the use of relative paths when importing a distro (#1613)
- Fix /etc/xinetd.d/rsync check (#1651)
- Exit with a appropiate message when signature file can't be parsed
- Handle cases where virt-install responds to `--version` on stderr (Koan)
- Fix mangling of kernel options in edit profile command with `--in-place`
- Several fixes to Koan regarding os-info-query and os-variants

1.2 Distribution Support

Cobbler currently supports importing a wide array of distributions from many vendors. However, since Cobbler has a history rooted in Red Hat based distributions, support for them is definitely the strongest. For others, the level of support varies from very good to requiring a lot of manual steps to get things working smoothly.

Here is the full list of supported distributions.

Note: This list does not include support for images, which can be just about any OS.

Key:

- X: Fully supported
- /: Some support, not 100%
- –: Not supported or siginificant manual action required

Family	Distro	Import	Import `--available-as`	PXE	Build ISO	Snippets
Red Hat	Fedora 16	X	X	X	X	X
	Fedora 17	X	X	X	X	X
	Fedora 18	X	X	X	X	X
	Fedora 19	X	X	X	X	X
	Fedora 20	X	X	X	X	X
	Fedora 21	X	X	X	X	X
	Fedora 22	X	X	X	X	X
	Fedora 23	X	X	X	X	X
	Fedora 24	X	X	X	X	X
	Fedora 25	X	X	X	X	X
	Fedora 26	X	X	X	X	X
	Fedora 27	X	X	X	X	X
	Fedora 28	X	X	X	X	X
	RHEL/CentOS 4	X	X	X	X	X
	RHEL/CentOS 5	X	X	X	X	X
	RHEL/CentOS 6	X	X	X	X	X
	RHEL/CentOS 7	X	X	X	X	X
Ubuntu	Lucid	X	–	X	X	/
	Oneiric	X	–	X	X	/
	Precise	X	–	X	X	/
	Quantal	X	–	X	X	/
	Raring	X	–	X	X	/
	Saucy	X	–	X	X	/
	Trusty	X	–	X	X	/
	Vivid	X	–	X	X	/
	Wily	X	–	X	X	/

Continued on next page

Table 1 – continued from previous page

Family	Distro	Import	Import `--available-as`	PXE	Build ISO	Snippets
	Xenial	X	–	X	X	/
	Yakkety	X	–	X	X	/
	Zesty	X	–	X	X	/
	Artful	X	–	X	X	/
	Bionic	X	–	X	X	/
Debian	Squeeze	X	–	/	X	/
	Wheezy	X	–	/	X	/
	Jessie	X	–	/	X	/
	Stretch	X	–	/	X	/
SUSE	openSUSE 11.2	X	X	X	X	/
	OpenSuSE 11.3	X	X	X	X	/
	OpenSuSE 11.4	X	X	X	X	/
	OpenSuSE 12.1	X	X	X	X	/
	OpenSuSE 12.2	X	X	X	X	/
	OpenSuSE 12.3	X	X	X	X	/
	OpenSuSE 13.1	X	X	X	X	/
	OpenSuSE 13.2	X	X	X	X	/
	SLES 10 sp4	X	X	X	X	/
	SLES 11	X	X	X	X	/
	SLES 11 sp1	X	X	X	X	/
	SLES 11 sp2	X	X	X	X	/
	SLES 11 sp3	X	X	X	X	/
	SLES 11 sp4	X	X	X	X	/
	SLES 12	X	X	X	X	/
	SLES 12 SP1	X	X	X	X	/
	SLES 12 SP2	X	X	X	X	/
VMware	ESX 4	X	–	X	X	/
	ESXi 4	X	–	X	–	–
	ESXi 5	X	–	/	–	–
	ESXi 5.1	X	–	/	–	–
	ESXi 5.5	X	–	/	–	–
	ESXi 6.0	X	–	/	–	–
	ESXi 6.5	X	–	/	–	–
FreeBSD	8.2	X	–	/	–	–
	8.3	X	–	/	–	–
	8.4	X	–	/	–	–
	9.0	X	–	/	–	–
	9.1	X	–	/	–	–
	9.2	X	–	/	–	–
	9.3	X	–	/	–	–
	10.0	X	–	/	–	–
	10.1	X	–	/	–	–
	10.2	X	–	/	–	–

Continued on next page

Table 1 – continued from previous page

Family	Distro	Import	Import `--available-as`	PXE	Build ISO	Snippets
	10.3	X	–	/	–	–
	11	X	–	/	–	–
Xen	XCP 1.6	X	–	/	–	–
	XenServer 6.2	X	–	/	–	–
	XenServer 6.5	X	–	/	–	–
	XenServer 7.0	X	–	/	–	–
	XenServer 7.1	X	–	/	–	–
Nexenta	Nexenta 4	X	–	/	–	–

1.3 Distribution Notes

Cobbler was originally written to support Fedora, Red Hat, and derivative distributions such as CentOS or Scientific Linux. Cobbler now works for managing other environments, including mixed environments, occasionally with some limitations. Debian/Ubuntu and SuSE support is quite strong, with patches coming in from developers working on those distributions as well.

However, in some cases (especially for newer distribution versions), a little extra work may be required after an import in order to make things work smoothly.

1.3.1 Nexenta

Installing NexentaStor with Cobbler

The following steps outline the Nexenta install process when using Cobbler.

1) Assuming that Cobbler has been setup previously, verify that the signature file contains the entry for Nexenta:

```
{
    "nexenta": {
        "4": {
            "signatures":["boot"],
            "version_file": "platform",
            "version_file_regex": null,
            "supported_arches":["x86_64"],
            "supported_repo_breeds":["apt"],
            "kernel_file":"platform/i86pc/kernel/amd64/unix",
            "initrd_file":"platform/i86pc/amd64/miniroot",
            "isolinux_ok":false,
            "kernel_options":"",
            "kernel_options_post":"",
            "boot_files":[]
        }
    }
}
```

2) Obtain a Nexenta iso from http://www.nexenta.com/corp/nexentastor-download and mount it:

```
mkdir -p /mnt/nexenta4 && mnt /path/to/nexenta4.iso /mnt/nexenta4 -o loop`
```

3) Import the distribution into Cobbler:

```
cobbler import --name=nexenta-4 --path=/mnt/nexenta4
```

Verify that a Nexenta distirbution is available via Cobbler: `cobbler list` Once the import is done, you can unmount the ISO:

```
sudo umount /mnt/nexenta4
```

4) Nexenta uses a PXE Grub executable different from other, linux-like systems. To install a Nexenta on a desired system, you have to specify the PXE Grub file for that system. This can be done by using either a MAC address, or a subnet definition in your DHCP configuration file. In `/etc/cobbler/dhcp.template`:

```
host test-1 {
    hardware ethernet 00:0C:29:10:B6:10;
    fixed-address 10.3.30.91;
    filename "boot/grub/pxegrub";
}
host test-2 {
    hardware ethernet 00:0c:29:d1:9c:26;
    fixed-address 10.3.30.97;
    filename "boot/grub/pxegrub";
}
```

OR if you are installing only Nexenta on all machines on a subnet, you may use the subnet definition instead of host definition in your dhcp config file.

Note: The path `boot/grub/pxegrub` is a hardcoded default in the Nexenta boot process.

---

5) In order the have unmanned installation, an installation profile must be created for each booted Nexenta system. The profiles are placed in `/var/lib/cobbler/kickstarts/install_profiles`. Each profile should be a file with the filename `machine.AACC003355FF` where AA..FF stand for the mac address of the machine, without *:* (columns). The contents of each profile should look like the following:

```
__PF_gateway="IP address" (required)
__PF_nic_primary="NIC NAME" (required)
__PF_dns_ip_1="IP address" (required)
__PF_dns_ip_2="IP address" (optional)
__PF_dns_ip_3="IP address" (optional)
__PF_loghost="IP address" (optional)
__PF_logport="Port Number" (optional)
__PF_syspool_luns="list of space separated LUNs that will be used to create␣
↪syspool" (required)
__PF_syspool_spare="list of space separated LUNs that will be used as syspool␣
↪spare" (optional)
```

(continues on next page)

(continued from previous page)

```
__PF_ipaddr_NIC_NAME="IP address" (NIC_NAME is the name of the target NIC␣
↪e1000g0, ixgbe1, etc.) (required)
__PF_netmask_NIC_NAME="NETMASK" (NIC_NAME is the name of the target NIC␣
↪e1000g0, ixgbe1, etc.) (required)
__PF_nlm_key="LICENSE KEY" (required)
__PF_language="en" (used to choose localzation, but now only "en" is␣
↪supported) (required)
__PF_ssh_enable=1 (enable SSH, by default SSH is disabled) (optional)
__PF_ssh_port="PORT where SSH server will wait for incoming connections"␣
↪(optional)
```

6) Power on the hardware. NexentaStor should boot from this setup.

Hints & Notes

This process has been tested with Cobbler Release 2.8.0 running on Ubuntu 12.04 LTS.

The install of Nexenta is automatic. That means that each machine to be booted with nexenta has to be configurated with a profile in kickstarts/install_profiles directory. To boot Nexenta nodes manually, in the file `/var/lib/tftpboot/boot/grub/menu.lst` replace the line:

```
kernel$ /images/nexenta-a-x86_64/platform/i86pc/kernel/amd64/unix -B iso_nfs_
↪path=10.3.30.95:/var/www/cobbler/links/nexenta-a-x86_64,auto_install=1
```

With

```
kernel$ /images/nexenta-a-x86_64/platform/i86pc/kernel/amd64/unix -B iso_nfs_
↪path=10.3.30.95:/var/www/cobbler/links/nexenta-a-x86_64
```

If you are adding a new distro, don't forget to enable NFS access to it! NFS share must be configured on the boot server. In particular, the directories in `/var/www/cobbler/links/<distro-name>` are exported. As an example, there is a `/etc/exports` file:

```
# /etc/exports: the access control list for filesystems which may be exported
#    to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_
↪subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/var/www/cobbler/links/nexenta-a-x86_64 *(ro,sync,no_subtree_check)
/var/www/cobbler/links/<nexenta-distribution-name> *(ro,sync,no_subtree_check)
```

1.3.2 FreeBSD

The following steps are required to enable FreeBSD support in Cobbler.

You can grab the patches and scripts from the following github repos: jsabo/cobbler_misc

This would not be possible without the help from Doug Kilpatrick. Thanks Doug!

Stuff to do once

- Install FreeBSD with full sources
  - Select "Standard" installation
  - Use entire disk
  - Install a standard MBR
  - Create a new slice and use the entire disk
  - Mount it at `/`
  - Choose the "Developer" distribution
    - Full sources, binaries and doc but no games
  - Install from a FreeBSD CD/DVD
  - Setup networking to copy files back and forth
  - In the post install "Package Selection" scroll down and select shells
    - Install bash
    - `chsh -s /usr/local/bin/bash username` or `vipw`
- Rebuild pxeboot with tftp support

```
cd /sys/boot
make clean
make LOADER_TFTP_SUPPORT=yes
make install
```

- Copy the pxeboot file to the Cobbler server.

Stuff to do every supported release

- Patch sysinstall with http install support
- The media location is hard coded in this patch and has to be updated every release. Just look for 8.X and change it. The standard sysinstall doesn't really support HTTP. This patch adds full http support to sysinstall.

```
cd /usr
patch -p0 < /root/http_install.patch
```

- Rebuild FreeBSD mfsroot: We'll use `crunchgen` to create the contents of /stand in a ramdisk image. Crunchgen creates a single statically linked binary that acts like different normal binaries depending on how it's called. We need to include `fetch` and a few other binaries. This is a multi step process.

```
mkdir /tmp/bootcrunch
cd /tmp/bootcrunch
crunchgen -o /root/boot_crunch.conf
make -f boot_crunch.mk
```

Once we've added our additional binaries we need to create a larger ramdisk.

- Create a new, larger ramdisk, and mount it.

```
dd if=/dev/zero of=/tmp/mfsroot bs=1024 count=$((1024 * 5))
dev0=`mdconfig -f /tmp/mfsroot`;newfs $dev0;mkdir /mnt/mfsroot_new;mount /dev/
↪$dev0 /mnt/mfsroot_new
```

- Mount the standard installer's mfsroot

```
mkdir /mnt/cdrom; mount -t cd9660 -o -e /dev/acd0 /mnt/cdrom
cp /mnt/cdrom/boot/mfsroot.gz /tmp/mfsroot.old.gz
gzip -d /tmp/mfsroot.old.gz; dev1=`mdconfig -f /tmp/mfsroot.old`
mkdir /mnt/mfsroot_old; mount /dev/$dev1 /mnt/mfsroot_old
```

Copy everything from the old one to the new one. You'll be replacing the binaries, but it's simpler to just copy it all over.

```
(cd /mnt/mfsroot_old/; tar -cf - .) | (cd /mnt/mfsroot_new; tar -xf -)
```

Next copy over the new bootcrunch file and create all of the symlinks after removing the old binaries.

```
cd /mnt/mfsroot_new/stand; rm -- *; cp /tmp/bootcrunch/boot_crunch ./
for i in $(./boot_crunch 2>&1|grep -v usage);do if [ "$i" != "boot_crunch" ];
↪then rm -f ./"$i";ln ./boot_crunch "$i";fi;done
```

Sysinstall uses install.cfg to start the install off. We've created a version of the install.cfg that uses fetch to pull down another configuration file from the Cobbler server which allows us to dynamically control the install. install.cfg uses a script called "doconfig.sh" to determine where the Cobbler installer is via the DHCP next-server field.

Copy both install.cfg and doconfig.sh into place.

```
cp {install.cfg,doconfig.sh} /mnt/mfsroot_new/stand
```

Now just unmount the ramdisk and compress the file

```
umount /mnt/mfsroot_new; umount /mnt/mfsroot_old
mdconfig -d -u $dev0; mdconfig -d -u $dev1
gzip /tmp/mfsroot
```

Copy the mfsroot.gz to the Cobbler server.

Stuff to do in Cobbler

- Enable Cobbler's tftp server in modules.conf

```
[tftpd]
module = manage_tftpd_py
```

- Mount the media

```
mount /dev/cdrom /mnt
```

- Import the distro

```
cobbler import --path=/mnt/ --name=freebsd-8.2-x86_64
```

- Copy the mfsroot.gz and the pxeboot.bs into the distro

```
cp pxeboot.bs /var/www/cobbler/ks_mirror/freebsd-8.2-x86_64/boot/
cp mfsroot.gz /var/www/cobbler/ks_mirror/freebsd-8.2-x86_64/boot/
```

- Configure a system to use the profile, turn on netboot, and off you go.

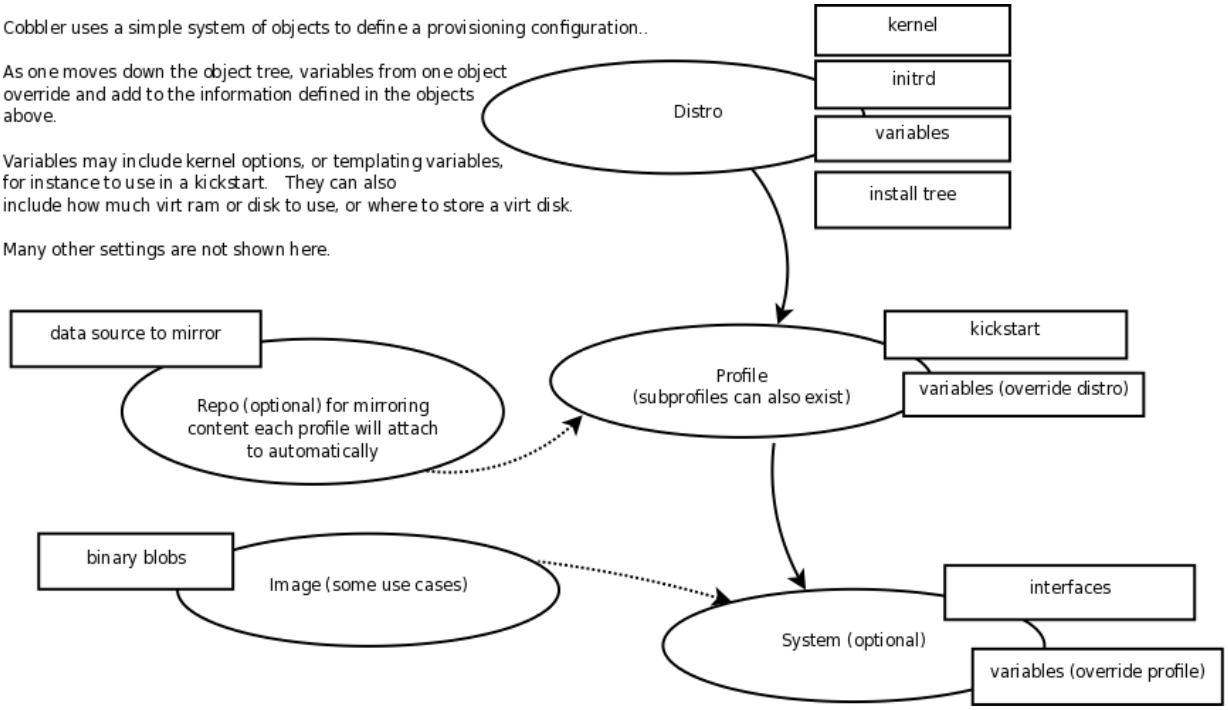
DHCP will tell the system to request `pxelinux.0`, so it will. Pxelinux will request it's configuration file, which will have `pxeboot.bs` as the "kernel". Pxelinux will request `pxeboot.bs`, use the extention (`.bs`) to realize it's another boot loader, and chain to it. Pxeboot will then request all the `.rc`, `.4th`, the kernel, and `mfsroot.gz`. It will mount the ramdisk and start the installer. The installer will connect back to the Cobbler server to fetch the `install.cfg` (the kickstart file), and do the install as instructed, rebooting at the end.

Cobbler is a build and deployment system. The primary functionality of cobbler is to simplify the lives of administrators by automating repetive actions, and to encourage reuse of existing work through the use of templating.

One of the primary tenets we follow is to provide options and flexibility rather than locking administrators into a single way of doing things. As such, cobbler can be integrated with a growing number of configuration management systems and remote scripting utilities while enabling deployment of many different operating system types.

Cobbler also provides a tool (koan) for simplifying virtualization deployments.

1.4 How We Model Things



CHAPTER 2

Installation

Cobbler is available for installation in several different ways, through packaging systems for each distribution or directly from source.

2.1 Prerequisites

Cobbler has both definite and optional prerequisites, based on the features you'd like to use. This section documents the definite prerequisites for both a basic installation and when building/installing from source. Please see the *Managing Services with Cobbler* section for details on the prerequisites for managing services, and other sections may make note of other package requirements.

Please note that installing any of the packages here via a package manager (such as yum or apt) can and will require a large number of ancilary packages, which we do not document here. The package definition should automatically pull these packages in and install them along with cobbler, however it is always best to verify these requirements have been met prior to installing cobbler or any of its components.

2.1.1 Cobbler/Cobblerd

First and foremost, cobbler requires Python. Any version over 2.6 should work. Cobbler also requires the installation of the following packages:

- createrepo
- httpd (apache2 for Debian/Ubuntu)
- mkisofs
- mod_wsgi (libapache2-mod-wsgi for Debian/Ubuntu)
- mod_ssl (libapache2-mod-ssl)

- python-cheetah
- python-netaddr
- python-simplejson
- python-urlgrabber
- PyYAML (python-yaml for Debian/Ubuntu)
- rsync
- syslinux
- tftp-server (atftpd for Debian/Ubuntu, though others _may_ work)
- yum-utils

2.1.2 Cobbler-Web

Cobbler web only has one other requirement besides cobbler itself:

- Django (python-django for Debian/Ubuntu)

2.1.3 Koan

Koan can be installed apart from cobblerd, and has only the following requirement (besides python itself of course):

- python-simplejson

2.1.4 Source Prerequisites

Installation from source requires the following additional packages:

- git
- make
- python-devel
- python-setuptools
- python-cheetah
- openssl

2.2 Installing from packages

Cobbler is available for installation for many Linux variants through their native packaging systems.

The Cobbler project also provides packages: http://cobbler.github.io/downloads/2.8.x.html

2.2.1 Fedora

Cobbler is packaged and available through the Fedora packaging system, so you just need to install the packages with the yum command: `sudo yum install cobbler`

With Fedora's packaging system, new releases are held in a "testing" repository for a period of time to vet bugs. If you would like to install the most up to date version of cobbler for Fedora (which may not be fully vetted for a production environment), enable the -testing repo when installing or updating:

```
$ sudo yum install --enablerepo=updates-testing cobbler
# or
$ sudo yum update --enablerepo=updates-testing cobbler
```

Once cobbler is installed, start and enable the service:

```
$ systemctl start cobblerd.service
$ systemctl enable cobblerd.service
$ systemctl status cobblerd.service
cobblerd.service - Cobbler Helper Daemon
      Loaded: loaded (/lib/systemd/system/cobblerd.service; enabled)
      Active: active (running) since Sun, 17 Jun 2012 13:01:28 -0500; 1min␣
↪44s ago
    Main PID: 1234 (cobblerd)
      CGroup: name=systemd:/system/cobblerd.service
          └ 1234 /usr/bin/python /usr/bin/cobblerd -F
```

And (re)start/enable Apache:

```
$ systemctl start httpd.service
$ systemctl enable httpd.service
```

2.2.2 RHEL and CentOS

Cobbler is packaged for RHEL variants through the Fedora EPEL (https://fedoraproject.org/wiki/EPEL) (Extra Packages for Enterprise Linux) system. Follow the directions there to install the correct repo RPM for your RHEL version and architecture. For example, on for a RHEL6.x x86_64 system:

```
$ sudo rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-
↪release-X-Y.noarch.rpm
```

Be sure to use the most recent X.Y version of the epel-release package. Once that is complete, simply use the yum command to install the cobbler package: `sudo yum install cobbler`

As noted above, new releases in the Fedora packaging system are held in a "testing" repository for a period of time to vet bugs. If you would like to install the most up to date version of cobbler through EPEL (which may not be fully vetted for a production environment), enable the -testing repo when installing or updating:

```
$ sudo yum install --enablerepo=epel-testing cobbler
# or
$ sudo yum update --enablerepo=epel-testing cobbler
```

Once cobbler is installed, start and enable the service:

```
$ service cobblerd start
$ chkconfig cobblerd on
```

And (re)start/enable Apache:

```
$ service httpd start
$ service cobblerd on
```

2.2.3 openSUSE

Enable required apache modules (`/etc/sysconfig/apache2:APACHE_MODULES`)

```
/usr/sbin/a2enmod proxy
/usr/sbin/a2enmod proxy_http
/usr/sbin/a2enmod proxy_connect
/usr/sbin/a2enmod rewrite
/usr/sbin/a2enmod ssl
/usr/sbin/a2enmod wsgi
/usr/sbin/a2enmod version
/usr/sbin/a2enmod socache_shmcb (or whatever module you are using)
```

Setup SSL certificates in Apache (not documented here)

Enable required apache flag (`/etc/sysconfig/apache2:APACHE_SERVER_FLAGS`)

```
/usr/sbin/a2enflag SSL
```

Make sure port 80 & 443 are opened in SuSEFirewall2 (not documented here)

Start/enable the apache2 and cobblerd services

```
systemctl enable apache2.service
systemctl enable cobblerd.service
systemctl start apache2.service
systemctl start cobblerd.service
```

Visit `https://${CERTIFICATE_FQDN}/cobbler_web/`

2.2.4 Debian and Ubuntu

TO BE DONE

2.3 Installing from Source

Cobbler is licensed under the General Public License (GPL), version 2 or later. You can download it, free of charge, using the links below.

2.3.1 Latest Source

The latest source code (it's all Python) is available through git (https://github.com/cobbler/cobbler).

Getting the Code

Clone the repo using git: git://github.com/cobbler/cobbler.git or https://github.com/cobbler/cobbler.git

```
$ git clone PASTE URL HERE
$ cd cobbler
$ git checkout release28
```

Note: The release28 branch corresponds to the official release version for the 2.8.x series. The master branch is the development series, and always uses an odd number for the minor version (for example, 2.9.0).

2.3.2 Installing

When building from source, make sure you have the correct *Prerequisites*. Once they are, you can install cobbler with the following command: `make install`

This command will rewrite all configuration files on your system if you have an existing installation of Cobbler (whether it was installed via packages or from an older source tree). To preserve your existing configuration files, snippets and kickstarts, run this command: `make devinstall`

To install the Cobbler web GUI, use this command: `make webtest`

Note: This will do a full install, not just the web GUI. `make webtest` is a wrapper around `make devinstall`, so your configuration files will also be saved when running this command.

Building Packages from Source (RPM)

It is also possible to build packages from the source file. Right now, only RPMs are supported, however we plan to add support for building .deb files in the future as well.

To build RPMs from source, use this command:

```
$ make rpms
... (lots of output) ...
Wrote: /path/to/cobbler/rpm-build/cobbler-2.8.0-1.fc24.src.rpm
Wrote: /path/to/cobbler/rpm-build/cobbler-2.8.0-1.fc24.noarch.rpm
Wrote: /path/to/cobbler/rpm-build/koan-2.8.0-1.fc24.noarch.rpm
Wrote: /path/to/cobbler/rpm-build/cobbler-web-2.8.0-1.fc24.noarch.rpm
```

As you can see, an RPM is output for each component of cobbler, as well as a source RPM. This command was run on a system running Fedora 20, thus the fc20 in the RPM name - this will be different based on the distribution you're running.

Building Packages from Source (DEB)

To install cobbler from source on Debian Squeeze, the following steps need to be made:

```
$ apt-get install make # for build
$ apt-get install git # for build
$ apt-get install python-yaml
$ apt-get install python-cheetah
$ apt-get install python-netaddr
$ apt-get install python-simplejson
$ apt-get install python-urlgrabber
$ apt-get install libapache2-mod-wsgi
$ apt-get install python-django
$ apt-get install atftpd

$ a2enmod proxy
$ a2enmod proxy_http
$ a2enmod rewrite

$ a2ensite cobbler.conf

$ ln -s /usr/local/lib/python2.6/dist-packages/cobbler /usr/lib/python2.6/
↪dist-packages/
$ ln -s /srv/tftp /var/lib/tftpboot

$ chown www-data /var/lib/cobbler/webui_sessions
```

- Change all `/var/www/cobbler` in `/etc/apache2/conf.d/cobbler.conf` to `/usr/share/cobbler/webroot/`
- init script - add Required-Stop line - path needs to be `/usr/local/...` or fix the install location

2.4 Configuration Files

Cobbler has many configuration files, however only a few typically need be modified for basic functionality:

2.4.1 Settings File

The main settings file for cobbler is `/etc/cobbler/settings`. Cobbler also supports *Dynamic Settings*, so it is no longer required to manually edit this file if this feature is enabled. This file is YAML-formatted, and with dynamic settings enabled [Augeas](http://augeas.net/) is used to modify its contents.

Whether dynamic settings are enabled or not, if you directly edit this file you must restart cobblerd. When modified with the dynamic settings CLI command or the web GUI, changes take affect immediately and do not require a restart.

2.4.2 Modules Configuration

Cobbler supports add-on modules, some of which can provide the same functionality (for instance, the authentication/authorization modules discussed in the *Web Authentication* section). Modules of this nature are configured via the `/etc/cobbler/modules.conf` file, for example:

```
# dns:
# chooses the DNS management engine if manage_dns is enabled
# in /etc/cobbler/settings, which is off by default.
# choices:
#    manage_bind    -- default, uses BIND/named
#    manage_dnsmasq -- uses dnsmasq, also must select dnsmasq for dhcp below

[dns]
module = manage_bind
```

As you can see above, this file has a typical INI-style syntax where sections are denoted with the *[&]* brackets and entries are of the form `key = value`.

Many of these sections are covered in the *Managing Services with Cobbler* and *Web Authentication* topics later in this manual. Please refer to those sections for further details on modifying this file.

As with the settings file, you must restart cobblerd after making changes to this file.

2.5 Relocating your installation

Often folks don't have a very large `/var` partition, which is what cobbler uses by default for mirroring install trees and the like.

You'll notice you can reconfigure the webdir location just by going into `/etc/cobbler/settings`, but it's not the best way to do things – especially as the RPM packaging does include some files and directories in the stock path. This means that, for upgrades and the like, you'll be breaking things somewhat. Rather than attempting to reconfigure cobbler, your Apache configuration, your file permissions, and your SELinux rules, the recommended course of action is very simple.

1. Copy everything you have already in `/var/www/cobbler` to another location – for instance, `/opt/cobbler_data`
2. Now just create a symlink or bind mount at `/var/www/cobbler` that points to `/opt/cobbler_data`.

Done. You're up and running.

Note: If you decided to access cobbler's data store over NFS (not recommended) you really want to mount NFS on <code>/var/www/cobbler</code> with SELinux context passed in as a parameter to mount versus the symlink. You may also have to deal with problems related to rootsquash. However if you are making a mirror of a Cobbler server for a multi-site setup, mounting read only is ok there.

Warning: `/var/lib/cobbler` can not live on NFS, as this interferes with locking ("flock") cobbler does around it's storage files.

CHAPTER 3

General

This section of the manual covers the basic functions of cobbler, which most people will use.

3.1 Cobbler Primitives

Primitives are the building blocks Cobbler uses to represent builds, as outlined in the "How We Model Things" section of the **:ref:'Introduction to Cobbler <about>'_** page. These objects are generally loosely related, though the distro/profile/system relation is somewhat more strict.

This section covers the creation and use of these objects, as well as how they relate to each other - including the methodology by which attributes are inherited from parent objects.

3.1.1 Standard Rules

Cobbler has a standard set of rules for manipulating primitive field values and, in the case of distros/profiles/systems, how those values are inherited from parents to children.

Inheritance of Values

Inheritance of values is based on the field type.

- For regular fields and arrays, the value will only be inherited if the field is set to `<<inherit>>`. Since distros and other objects like repos do not have a parent, these values are inherited from the defaults in *Cobbler Settings*. If the field is specifically set to an empty string, no value will be inherited.
- For hashes, the values from the parent will always be inherited and blended with the child values. If the parent and child have the same key, the child's values will win an override the parent's.

Array Fields

Some fields in Cobbler (for example, the `--name-servers` field) are stored as arrays. These arrays are always considered arrays of strings, and are always specified in Cobbler as a space-separated list when using add/edit.

Example:

```
$ cobbler [object] edit --name=foo --field="a b c d"
```

Hash Fields (key=value)

Other fields in Cobbler (for example, the –ksmeta field) are stored as hashes - that is a list of key=value pairs. As with arrays, both the keys and values are always interpreted as strings.

Preserving Values When Editing

By default, any time a hash field is manipulated during an edit, the contents of the field are replaced completely with the new values specified during the edit.

Example:

```
$ cobbler distro edit --name=foo --ksmeta="a=b c=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd'}
$ cobbler distro edit --name=foo --ksmeta="e=f"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'e': 'f'}
```

To preserve the contents of these fields, –in-place should be specified:

```
$ cobbler distro edit --name=foo --ksmeta="a=b c=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd'}
$ cobbler distro edit --name=foo --in-place --ksmeta="e=f"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd', 'e': 'f'}
```

Removing Values

To remove a single value from the hash, use the '~' (tilde) character along with –in-place:

```
$ cobbler distro edit --name=foo --ksmeta="a=b c=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd'}
$ cobbler distro edit --name=foo --in-place --ksmeta='~a'
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'c': 'd'}
```

Suppressing Values

You can also suppress values from being used, by specifying the '-' character in front of the key name:

```
$ cobbler distro edit --name=foo --ksmeta="a=b c=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd'}
$ cobbler distro edit --name=foo --in-place --ksmeta='-a'
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'-a': 'b', 'c': 'd'}
```

In this case, the key=value pair will be ignored when the field is accessed.

Keys Without Values

You can always specify keys without a value:

```
$ cobbler distro edit --name=foo --ksmeta="a b c"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': '~', 'c': '~', 'b': '~'}
```

Note: While valid syntax, this could cause problems for some fields where Cobbler expects a value (for example, –template-files).

Keys With Multiple Values

It is also possible to specify multiple values for the same key. In this situation, Cobbler will convert the value portion to an array:

```
$ cobbler distro edit --name=foo --in-place --ksmeta="a=b a=c a=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': ['b', 'c', 'd']}
```

Note: You must specify `--in-place` for this to work. By default the behavior will result in a single value, with the last specified value being the winner.

3.1.2 Standard Primitive Sub-commands

All primitive objects support the following standard sub-commands:

List

The list command simply prints out an alphabetically sorted list of all objects.

Example:

```
$ cobbler distro list
   centos6.3-x86_64
   debian6.0.5-x86_64
   f17-x86_64
   f18-beta6-x86_64
   opensuse12.2-i386
   opensuse12.2-x86_64
   opensuse12.2-xen-i386
   opensuse12.2-xen-x86_64
   sl6.2-i386
   sl6.2-x86_64
   ubuntu-12.10-i386
   ubuntu-12.10-x86_64
```

The list command is actually available as a top-level command as well, in which case it will iterate through every object type and list everything currently stored in your Cobbler database.

Report

The report command prints a formatted report of each objects configuration. The optional `--name` argument can be used to limit the output to a single object, otherwise a report will be printed out for every object (if you have a lot of objects in a given category, this can be somewhat slow).

As with the list command, the report command is also available as a top-level command, in which case it will print a report for every object that is stored in your Cobbler database.

Remove

The remove command uses only the `--name` option.

Note: Removing an object will also remove any child objects (profiles, sub-profiles and/or systems). Prior versions of Cobbler required an additional `--recursive` option to enable this behavior, but it has become the default in recent versions so use remove with caution.

Example:

```
$ cobbler [object] remove --name=foo
```

Copy/Rename

The copy and rename commands work similarly, with both requiring a `--name` and `--newname` options.

Example:

```
$ cobbler [object] copy --name=foo --newname=bar
# or
$ cobbler [object] rename --name=foo --newname=bar
```

Find

The find command allows you to search for objects based on object attributes.

Please refer to the *Command Line Search* section for more details regarding the find sub-command.

Dumpvars (Debugging)

The dumpvars command is intended to be used for debugging purposes, and for those writing snippets. In general, it is not required for day-to-day use.

3.1.3 Cobbler Objects

Distros

The first step towards installing systems with Cobbler is to add a distribution record to cobbler's configuration.

The distro command has the following sub-commands:

```
$ cobbler distro --help
usage
=====
cobbler distro add
cobbler distro copy
cobbler distro edit
cobbler distro find
cobbler distro list
cobbler distro remove
cobbler distro rename
cobbler distro report
```

Add/Edit Options

In general, it's really a lot easier to follow the import workflow – it only requires waiting for the mirror content to be copied and/or scanned. Imported mirrors also save time during install since they don't have to hit external installation sources. Please read the *Import* documentation for more details.

If you want to be explicit with distribution definition, however, here's how it works:

Example:

```
$ cobbler distro add --name=string --kernel=path --initrd=path [options]
```

–name (required)

A string identifying the distribution, this should be something like "rhel4".

–kernel (required)

An absolute filesystem path to a kernel image.

–initrd (required)

An absolute filesystem path to a initrd image.

–arch

Sets the architecture for the PXE bootloader and also controls how koan's `--replace-self` option will operate.

The default setting ('standard') will use pxelinux. Set to 'ia64' to use elilo. 'ppc' and 'ppc64' use yaboot. 's390x' is not PXEable, but koan supports it for reinstalls.

'x86' and 'x86_64' effectively do the same thing as standard.

If you perform a cobbler import, the arch field will be auto-assigned.

–boot-files

This option is used to specify additional files that should be copied to the TFTP directory for the distro so that they can be fetched during earlier stages of the installation. Some distributions (for example, VMware ESXi) require this option to function correctly.

–breed

Controls how various physical and virtual parameters, including kernel arguments for automatic installation, are to be treated. Defaults to "redhat", which is a suitable value for Fedora and CentOS as well. It means anything redhat based.

There is limited experimental support for specifying "debian", "ubuntu", or "suse", which treats the kickstart file as a different format and changes the kernel arguments appropriately. Support for other types of distributions is possible in the future. See the Wiki for the latest information about support for these distributions.

The file used for the answer file, regardless of the breed setting, is the value used for `--kickstart` when creating the profile. In other words, if another distro calls their answer file something other than a "kickstart", the kickstart parameter still governs where that answer file is.

–clobber

This option allows "add" to overwrite an existing distro with the same name, so use it with caution.

–comment

An optional comment to associate with this distro.

–fetchable-files

This option is used to specify a list of key=value files that can be fetched via the python based TFTP server. The "value" portion of the name is the path/name they will be available as via TFTP.

Please see the *TFTP* section for more details on using the python-based TFTP server.

–in-place

By default, any modifications to key=value fields (ksmeta, kopts, etc.) do no preserve the contents.

Example:

```
$ cobbler distro edit --name=foo --ksmeta="a=b c=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd'}
$ cobbler distro edit --name=foo --ksmeta="e=f"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'e': 'f'}
```

To preserve the contents of these fields, `--in-place` should be specified:

```
$ cobbler distro edit --name=foo --ksmeta="a=b c=d"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd'}
$ cobbler distro edit --name=foo --in-place --ksmeta="e=f"
$ cobbler distro report --name=foo | grep "Kickstart Meta"
Kickstart Metadata             : {'a': 'b', 'c': 'd', 'e': 'f'}
```

–kopts

Sets kernel command-line arguments that the distro, and profiles/systems dependant on it, will use during installation only. This field is a hash field, and accepts a set of key=value pairs:

Example:

```
--kopts="console=tty0 console=ttyS0,8,n,1 noapic"
```

–kopts-post

This is just like `--kopts`, though it governs kernel options on the installed OS, as opposed to kernel options fed to the installer. This requires some special snippets to be found in your kickstart template to work correctly.

–ksmeta

This is an advanced feature that sets variables available for use in templates. This field is a hash field, and accepts a set of key=value pairs:

Example:

```
--ksmeta="foo=bar baz=3 asdf"
```

See the section on *Kickstart Templating* for further information.

–mgmt-classes

Management classes that should be associated with this distro for use with configuration management systems.

Please see the *Configuration Management* section for more details on integrating Cobbler with configuration management systems.

–os-version

Generally this field can be ignored. It is intended to alter some hardware setup for virtualized instances when provisioning guests with koan. The valid options for `--os-version` vary depending on what is specified for `--breed`. If you specify an invalid option, the error message will contain a list of valid os versions that can be used. If you do not know the os version or it does not appear in the list, omitting this argument or using "other" should be perfectly fine. Largely this is needed to support older distributions in virtualized settings, such as "rhel2.1", one of the OS choices if the breed is set to "redhat". If you do not encounter any problems with virtualized instances, this option can be safely ignored.

–owners

The value for `--owners` is a space seperated list of users and groups as specified in `/etc/cobbler/users.conf`.

Users with small sites and a limited number of admins can probably ignore this option, since it only applies to the Cobbler WebUI and XMLRPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected when using the `authz_ownership` module which must be enabled and is not the default.

Please see the *Web Authorization* section for more details.

–redhat-management-key

If you're using Red Hat Network, Red Hat Satellite Server, or Spacewalk, you can store your authentication keys here and Cobbler can add the neccessary authentication code to your kickstart where the snippet named

`redhat_register` is included. The default option specified in *Cobbler Settings* will be used if this field is left blank.

Please see the *Tips for RHN* section for more details on integrating Cobbler with RHN/Spacewalk.

–redhat-management-server

The RHN Satellite or Spacewalk server to use for registration. As above, the default option specified in *Cobbler Settings* will be used if this field is left blank.

Please see the *Tips for RHN* section for more details on integrating Cobbler with RHN/Spacewalk.

–template-files

This feature allows cobbler to be used as a configuration management system. The argument is a space delimited string of key=value pairs. Each key is the path to a template file, each value is the path to install the file on the system. Koan also can retrieve these files from a cobbler server on demand, effectively allowing cobbler to function as a lightweight templated configuration management system.

Please see the *Built-In Configuration Management* section for more details on using template files.

Profiles and Sub Profiles

A profile associates a distribution to additional specialized options, such as a kickstart automation file. Profiles are the core unit of provisioning and at least one profile must exist for every distribution to be provisioned. A profile might represent, for instance, a web server or desktop configuration. In this way, profiles define a role to be performed.

The profile command has the following sub-commands:

```
$ cobbler profile --help
usage
=====
cobbler profile add
cobbler profile copy
cobbler profile dumpvars
cobbler profile edit
cobbler profile find
cobbler profile getks
cobbler profile list
cobbler profile remove
cobbler profile rename
cobbler profile report
```

Add/Edit Options

Example:

```
$ cobbler profile add --name=string --distro=string [options]
```

–name (required)

A descriptive name. This could be something like "rhel5webservers" or "f9desktops".

–distro (required)

The name of a previously defined cobbler distribution. This value is required.

–boot-files

This option is used to specify additional files that should be copied to the TFTP directory for the distro so that they can be fetched during earlier stages of the installation. Some distributions (for example, VMware ESXi) require this option to function correctly.

–clobber

This option allows "add" to overwrite an existing profile with the same name, so use it with caution.

–comment

An optional comment to associate with this profile.

–dhcp-tag

DHCP tags are used in the dhcp.template when using multiple networks.

Please refer to the *DHCP* section for more details.

–enable-gpxe

When enabled, the system will use gPXE instead of regular PXE for booting.

Please refer to the *Using gPXE* section for details on using gPXE for booting over a network.

–enable-menu

When managing TFTP, Cobbler writes the `${tftproot}/pxelinux.cfg/default` file, which contains entries for all profiles. When this option is enabled for a given profile, it will not be added to the default menu.

–fetchable-files

This option is used to specify a list of key=value files that can be fetched via the python based TFTP server. The "value" portion of the name is the path/name they will be available as via TFTP.

Please see the *TFTP* section for more details on using the python-based TFTP server.

–in-place

By default, any modifications to key=value fields (ksmeta, kopts, etc.) do no preserve the contents. To preserve the contents of these fields, `--in-place` should be specified. This option is also required is using a key with multiple values (for example, "foo=bar foo=baz").

–kickstart

Local filesystem path to a kickstart file. `http://` URLs (even CGI's) are also accepted, but a local file path is recommended, so that the kickstart templating engine can be taken advantage of.

If this parameter is not provided, the kickstart file will default to `/var/lib/cobbler/kickstarts/default.ks`. This file is initially blank, meaning default kickstarts are not automated "out of the box". Admins can change the `default.ks` if they desire.

When using kickstart files, they can be placed anywhere on the filesystem, but the recommended path is `/var/lib/cobbler/kickstarts`. If using the webapp to create new kickstarts, this is where the web application will put them.

–kopts

Sets kernel command-line arguments that the profile, and sub-profiles/systems dependant on it, will use during installation only. This field is a hash field, and accepts a set of key=value pairs.

Example:

```
--kopts="console=tty0 console=ttyS0,8,n,1 noapic"
```

–kopts-post

This is just like `--kopts`, though it governs kernel options on the installed OS, as opposed to kernel options fed to the installer. This requires some special snippets to be found in your kickstart template to work correctly.

–ksmeta

This is an advanced feature that sets variables available for use in templates. This field is a hash field, and accepts a set of key=value pairs:

Example:

```
--ksmeta="foo=bar baz=3 asdf"
```

See the section on *Kickstart Templating* for further information.

–mgmt-classes, –mgmt-parameters

Management classes and parameters that should be associated with this profile for use with configuration management systems.

Please see the *Configuration Management* section for more details on integrating Cobbler with configuration management systems.

–name-servers

If your nameservers are not provided by DHCP, you can specify a space seperated list of addresses here to configure each of the installed nodes to use them (provided the kickstarts used are installed on a per-system basis). Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.

–name-servers-search

As with the `--name-servers` option, this can be used to specify the default domain search line. Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.

–owners

The value for `--owners` is a space seperated list of users and groups as specified in `/etc/cobbler/users.conf`

Users with small sites and a limited number of admins can probably ignore this option, since it only applies to the Cobbler WebUI and XMLRPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected when using the `authz_ownership` module which must be enabled and is not the default.

Please see the *Web Authorization* section for more details.

–parent

This is an advanced feature.

Profiles may inherit from other profiles in lieu of specifing `--distro`. Inherited profiles will override any settings specified in their parent, with the exception of `--ksmeta` (templating) and `--kopts` (kernel options), which will be blended together.

Example: If profile A has `--kopts="x=7 y=2"`, B inherits from A, and B has `--kopts="x=9 z=2"`, the actual kernel options that will be used for B are `"x=9 y=2 z=2".</p>`

Example: If profile B has `--virt-ram=256` and A has `--virt-ram=512`, profile B will use the value 256.

Example: If profile A has a `--virt-file-size=5` and B does not specify a size, B will use the value from A.

–proxy

Specifies a proxy to use during the installation stage.

Note: Not all distributions support using a proxy in this manner.

–redhat-management-key

If you're using Red Hat Network, Red Hat Satellite Server, or Spacewalk, you can store your authentication keys here and Cobbler can add the neccessary authentication code to your kickstart where the snippet named `redhat_register` is included. The default option specified in *Cobbler Settings* will be used if this field is left blank.

Please see the *Tips for RHN* section for more details on integrating Cobbler with RHN/Spacewalk.

–redhat-management-server

The RHN Satellite or Spacewalk server to use for registration. As above, the default option specified in *Cobbler Settings* will be used if this field is left blank.

Please see the *Tips for RHN* section for more details on integrating Cobbler with RHN/Spacewalk.

–repos

This is a space delimited list of all the repos (created with `cobbler repo add` and updated with `cobbler reposync`) that this profile can make use of during kickstart installation. For example, an example might be `--repos="fc6i386updates fc6i386extras"` if the profile wants to access these two mirrors that are already mirrored on the cobbler server. Repo management is described in greater depth later in the manpage.

–server

This parameter should be useful only in select circumstances. If machines are on a subnet that cannot access the cobbler server using the name/IP as configured in the cobbler settings file, use this parameter to override

that server name. See also `--dhcp-tag` for configuring the next server and DHCP informmation of the system if you are also using Cobbler to help manage your DHCP configuration.

–template-files

This feature allows cobbler to be used as a configuration management system. The argument is a space delimited string of key=value pairs. Each key is the path to a template file, each value is the path to install the file on the system. Koan also can retrieve these files from a cobbler server on demand, effectively allowing cobbler to function as a lightweight templated configuration management system.

Please see the *Built-In Configuration Management* section for more details on using template files.

–template-remote-kickstarts

If enabled, any kickstart with a remote path (`http://`, `ftp://`, etc.) will not be passed through Cobbler's template engine.

–virt-auto-boot

(Virt-only) When set, the VM will be configured to automatically start when the host reboots.

–virt-bridge

(Virt-only) This specifies the default bridge to use for all systems defined under this profile. If not specified, it will assume the default value in the cobbler settings file, which as shipped in the RPM is 'xenbr0'. If using KVM, this is most likely not correct. You may want to override this setting in the system object. Bridge settings are important as they define how outside networking will reach the guest. For more information on bridge setup, see the Cobbler Wiki, where there is a section describing koan usage.

–virt-cpus

(Virt-only) How many virtual CPUs should koan give the virtual machine? The default for this value is set in the *Cobbler Settings* file, and should be set as an integer.

–virt-disk-driver

(Virt-only) The type of disk driver to use for the disk image, for example "raw" or "qcow2".

–virt-file-size

(Virt-only) How large the disk image should be in Gigabytes. The default for this value is set in the *Cobbler Settings* file. This can be a space seperated list (ex: "5,6,7") to allow for multiple disks of different sizes

depending on what is given to `--virt-path`. This should be input as a integer or decimal value without units.

–virt-path

(Virt-only) Where to store the virtual image on the host system. Except for advanced cases, this parameter can usually be omitted. For disk images, the value is usually an absolute path to an existing directory with an optional file name component. There is support for specifying partitions `/dev/sda4` or volume groups `VolGroup00`, etc.

For multiple disks, seperate the values with commas such as `VolGroup00,VolGroup00` or `/dev/sda4,/dev/sda5`. Both those examples would create two disks for the VM.

–virt-ram

(Virt-only) How many megabytes of RAM to consume. The default for this value is set in the *Cobbler Settings* file. This should be input as an integer without units, and will be interpretted as MB.

–virt-type

(Virt-only) Koan can install images using several different virutalization types. Choose one or the other strings to specify, or values will default to attempting to find a compatible installation type on the client system ("auto"). See the https://koan.readthedocs.io/ section for more documentation. The default for this value is set in the *Cobbler Settings* file.</td>

Get Kickstart (getks)

The getks command shows the rendered kickstart/response file (preseed, etc.) for the given profile. This is useful for previewing what will be downloaded from Cobbler when the system is building. This is also a good opportunity to catch snippets that are not rendering correctly.

As with remove, the `--name` option is required and is the only valid argument.

Example:

```
$ cobbler profile getks --name=foo | less
```

Systems

System records map a piece of hardware (or a virtual machine) with the cobbler profile to be assigned to run on it. This may be thought of as chosing a role for a specific system.

The system commmand has the following sub-commands:

```
$ cobbler system --help
usage
=====
cobbler system add
cobbler system copy
cobbler system dumpvars
cobbler system edit
cobbler system find
cobbler system getks
cobbler system list
cobbler system poweroff
cobbler system poweron
cobbler system powerstatus
cobbler system reboot
cobbler system remove
cobbler system rename
cobbler system report
```

Note that if provisioning via koan and PXE menus alone, it is not required to create system records in cobbler, though they are useful when system specific customizations are required. One such customization would be defining the MAC address. If there is a specific role inteded for a given machine, system records should be created for it.

System commands have a wider variety of control offered over network details. In order to use these to the fullest possible extent, the kickstart template used by cobbler must contain certain kickstart snippets (sections of code specifically written for Cobbler to make these values become reality). Compare your kickstart templates with the stock ones in `/var/lib/cobbler/kickstarts` if you have upgraded, to make sure you can take advantage of all options to their fullest potential. If you are a new cobbler user, base your kickstarts off of these templates. Non-kickstart based distributions, while supported by Cobbler, may not be able to use all of these features.

Example:

```
$ cobbler system add --name=string [--profile=name|--image=name] [options]
```

As you can see, a system must either be assigned to a `--profile` or an `--image`, which are mutually exclusive options.

Add/Edit Options

–name (required)

The system name works like the name option for other commands.

If the name looks like a MAC address or an IP, the name will implicitly be used for either `--mac` or `--ip-address` of the first interface, respectively. However, it's usually better to give a descriptive name – don't rely on this behavior.

A system created with name "default" has special semantics. If a default system object exists, it sets all undefined systems to PXE to a specific profile. Without a "default" system name created, PXE will fall

through to local boot for unconfigured systems.

When using "default" name, don't specify any other arguments than `--profile ...` they won't be used.

–profile (required, if –image not set)

The name of the profile or sub-profile to which this system belongs.

–image (required, if –profile not set)

The name of the image to which this system belongs.

–boot-files

This option is used to specify additional files that should be copied to the TFTP directory for the distro so that they can be fetched during earlier stages of the installation. Some distributions (for example, VMware ESXi) require this option to function correctly.

–clobber

This option allows "add" to overwrite an existing system with the same name, so use it with caution.

–comment

An optional comment to associate with this system.

–enable-gpxe

When enabled, the system will use gPXE instead of regular PXE for booting.

Please refer to the *Using gPXE* section for details on using gPXE for booting over a network.

–fetchable-files

This option is used to specify a list of `key=value` files that can be fetched via the python based TFTP server. The "value" portion of the name is the path/name they will be available as via TFTP.

Please see the *TFTP* section for more details on using the python-based TFTP server.

–gateway

Sets the default gateway, which in Redhat-based systems is typically in `/etc/sysconfig/network`. Per-interface gateways are not supported at this time. This option will be ignored unless `--static=1` is also set on the interface.

–hostname

This field corresponds to the hostname set in a systems `/etc/sysconfig/network` file. This has no bearing on DNS, even when `manage_dns` is enabled. Use `--dns-name` instead for that feature, which is a per-interface setting.

–in-place

By default, any modifications to `key=value` fields (`ksmeta`, `kopts`, etc.) do no preserve the contents. To preserve the contents of these fields, `--in-place` should be specified. This option is also required is using a key with multiple values (for example, `foo=bar foo=baz`).

–kickstart

While it is recommended that the `--kickstart` parameter is only used within for the "profile add" command, there are limited scenarios when an install base switching to cobbler may have legacy kickstarts created on a per-system basis (one kickstart for each system, nothing shared) and may not want to immediately make use of the cobbler templating system. This allows specifing a kickstart for use on a per-system basis. Creation of a parent profile is still required. If the kickstart is a filesystem location, it will still be treated as a cobbler template.

–kopts

Sets kernel command-line arguments that the system will use during installation only. This field is a hash field, and accepts a set of key=value pairs:

Example:

```
--kopts="console=tty0 console=ttyS0,8,n,1 noapic"
```

–kopts-post

This is just like `--kopts`, though it governs kernel options on the installed OS, as opposed to kernel options fed to the installer. This requires some special snippets to be found in your kickstart template to work correctly.

–ksmeta

This is an advanced feature that sets variables available for use in templates. This field is a hash field, and accepts a set of `key=value` pairs:

Example:

```
--ksmeta="foo=bar baz=3 asdf"
```

See the section on *Kickstart Templating* for further information.

–ldap-enabled, –ldap-type

Cobbler contains features that enable ldap management for easier configuration after system provisioning. If set true, koan will run the ldap command as defined by the systems ldap_type. The default value is false.

–mgmt-classes and –mgmt-parameters

Management classes and parameters that should be associated with this system for use with configuration management systems.

Please see the *Configuration Management* section for more details on integrating Cobbler with configuration management systems.

–monit-enabled

> **Warning:** This feature has been deprecated and will not be available in cobbler 3.0

If set true, koan will reload monit after each configuration run. The default value is false.

–name-servers

If your nameservers are not provided by DHCP, you can specify a space seperated list of addresses here to configure each of the installed nodes to use them (provided the kickstarts used are installed on a per-system basis). Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.

–name-servers-search

As with the `--name-servers` option, this can be used to specify the default domain search line. Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.

–netboot-enabled

If set false, the system will be provisionable through koan but not through standard PXE. This will allow the system to fall back to default PXE boot behavior without deleting the cobbler system object. The default value allows PXE. Cobbler contains a PXE boot loop prevention feature (`pxe_just_once`, can be enabled in `/etc/cobbler/settings`) that can automatically trip off this value after a system gets done installing. This can prevent installs from appearing in an endless loop when the system is set to PXE first in the BIOS order.

–owners

The value for `--owners` is a space seperated list of users and groups as specified in `/etc/cobbler/users.conf`.

–power-address, –power-type, –power-user, –power-password, –power-id

Cobbler contains features that enable integration with power management for easier installation, reinstallation, and management of machines in a datacenter environment. These parameters are described in the *Power Management* section under *Advanced*. If you have a power-managed datacenter/lab setup, usage of these features may be something you are interested in.

–proxy

Specifies a proxy to use during the installation stage.

Note: Not all distributions support using a proxy in this manner.

–redhat-management-key

If you're using Red Hat Network, Red Hat Satellite Server, or Spacewalk, you can store your authentication keys here and Cobbler can add the neccessary authentication code to your kickstart where the snippet named "redhat_register" is included. The default option specified in *Cobbler Settings* will be used if this field is left blank.

Please see the *Tips for RHN* section for more details on integrating Cobbler with RHN/Spacewalk.

–redhat-management-server

The RHN Satellite or Spacewalk server to use for registration. As above, the default option specified in *Cobbler Settings* will be used if this field is left blank.

Please see the *Tips for RHN* section for more details on integrating Cobbler with RHN/Spacewalk.

–repos-enabled

If set true, koan can reconfigure repositories after installation.

–server

This parameter should be useful only in select circumstances. If machines are on a subnet that cannot access the cobbler server using the name/IP as configured in the cobbler settings file, use this parameter to override that server name. See also `--dhcp-tag` for configuring the next server and DHCP informmation of the system if you are also using Cobbler to help manage your DHCP configuration.

–status

An optional field used to keep track of a systems build or deployment status. This field is only set manually, and is not updated automatically at this time.

–template-files

This feature allows cobbler to be used as a configuration management system. The argument is a space delimited string of key=value pairs. Each key is the path to a template file, each value is the path to install the file on the system. Koan also can retrieve these files from a cobbler server on demand, effectively allowing cobbler to function as a lightweight templated configuration management system.

Please see the *Built-In Configuration Management* section for more details on using template files.

–template-remote-kickstarts

If enabled, any kickstart with a remote path (`http://`, `ftp://`, etc.) will not be passed through Cobbler's template engine.

–virt-auto-boot

(Virt-only) When set, the VM will be configured to automatically start when the host reboots.

–virt-cpus

(Virt-only) The number of virtual CPUs to allocate to a system. The default for this value is set in the *Cobbler Settings* file, and should be set as an integer.

–virt-disk-driver

(Virt-only) The type of disk driver to use for the disk image, for example "raw" or "qcow2".

–virt-file-size

(Virt-only) How large the disk image should be in Gigabytes. The default for this value is set in the *Cobbler Settings* file. This can be a space seperated list (ex: "5,6,7") to allow for multiple disks of different sizes depending on what is given to *–virt-path*. This should be input as a integer or decimal value without units.

–virt-path

(Virt-only) Where to store the virtual image on the host system. Except for advanced cases, this parameter can usually be omitted. For disk images, the value is usually an absolute path to an existing directory with an optional file name component. There is support for specifying partitions `/dev/sda4` or volume groups `VolGroup00`, etc.

For multiple disks, seperate the values with commas such as `VolGroup00,VolGroup00` or `/dev/sda4,/dev/sda5`. Both those examples would create two disks for the VM.

–virt-pxe-boot

(Virt-only) When set, the guest VM will use PXE to boot. By default, koan will use the `--location` option to virt-install to specify the installer for the guest.

–virt-ram

(Virt-only) How many megabytes of RAM to consume. The default for this value is set in the *Cobbler Settings* file. This should be input as an integer without units, and will be interpretted as `MB`.

–virt-type

(Virt-only) Koan can install images using several different virutalization types. Choose one or the other strings to specify, or values will default to attempting to find a compatible installation type on the client system ("auto"). See the https://koan.readthedocs.io/ section for more documentation. The default for this value is set in the *Cobbler Settings* file.

Interface Specific Commands

System primitives are unique in that they are the only object in Cobbler that embeds another complex object - interfaces. As such, there is an entire subset of options that are specific to interfaces only.

–interface

All interface options require the use of the `--interface=ifname` option. If this is omitted, Cobbler will default to using the interface name "eth0", which may not be what you want. We may also change this

default behavior in the future, so in general it is always best to explicitly specify the interface name with this option.

Note: You can only edit one interface at a time! If you specify multiple `--interface` options, only the last one will be used.

Interface naming notes:

Additional interfaces can be specified (for example: eth1, or any name you like, as long as it does not conflict with any reserved names such as kernel module names) for use with the edit command. Defining VLANs this way is also supported, if you want to add VLAN 5 on interface eth0, simply name your interface eth0:5.

Example:

```
$ cobbler system edit --name=foo --ip-address=192.168.1.50 --
↪mac=AA:BB:CC:DD:EE:A0
$ cobbler system edit --name=foo --interface=eth0 --ip-address=192.168.1.51 --
↪mac=AA:BB:CC:DD:EE:A1
$ cobbler system report foo
```

Interfaces can be deleted using the `--delete-interface` option.

Example:

```
$ cobbler system edit --name=foo --interface=eth2 --delete-interface
```

–bonding-opts and –bridge-opts

Bonding and bridge options for the master-interface may be specified using –bonding-opts="foo=1 bar=2" or *–bridge-opts="foo=1 bar=2"*, respectively. These are only used if the *–interface-type* is a master or bonded_bridge_slave (which is also a bond master).

–dhcp-tag If you are setting up a PXE environment with multiple subnets/gateways, and are using cobbler to manage a DHCP configuration, you will probably want to use this option. If not, it can be ignored.

By default, the dhcp tag for all systems is "default" and means that in the DHCP template files the systems will expand out where $insert_cobbler_systems_definitions is found in the DHCP template. However, you may want certain systems to expand out in other places in the DHCP config file. Setting –dhcp-tag=subnet2 for instance, will cause that system to expand out where $insert_cobbler_system_definitions_subnet2 is found, allowing you to insert directives to specify different subnets (or other parameters) before the DHCP configuration entries for those particular systems.

–dns-name

If using the DNS management feature (see advanced section – cobbler supports auto-setup of BIND and dnsmasq), use this to define a hostname for the system to receive from DNS.

Example:

```
--dns-name=mycomputer.example.com
```

This is a per-interface parameter. If you have multiple interfaces, it may be different for each interface, for example, assume a DMZ/dual-homed setup.

–interface-type and –interface-master

One of the other advanced networking features supported by Cobbler is NIC bonding and bridging. You can use this to bond multiple physical network interfaces to one single logical interface to reduce single points of failure in your network, or to create bridged interfaces for things like tunnels and virtual machine networks. Supported values for the `--interface-type` parameter are "bond", "bond_slave", "bridge", "bridge_slave" and "bonded_bridge_slave". If one of the `_slave` options is specified, you also need to define the master-interface for this bond using `--interface-master=INTERFACE`.

Note: The options `master` and `slave` are deprecated, and are assumed to me `bond` and `bond_slave` when encountered. When a system object is saved, the deprecated values will be overwritten with the new, correct values.

For more details on using these interface types, please see the *Advanced Networking* section.

–ip-address

If cobbler is configured to generate a DHCP configuratition (see advanced section), use this setting to define a specific IP for this system in DHCP. Leaving off this parameter will result in no DHCP management for this particular system.

Example:

```
--ip-address=192.168.1.50
```

Note for Itanium users: This setting is always required for IA64 regardless of whether DHCP management is enabled.

If DHCP management is disabled and the interface is labelled `--static=1`, this setting will be used for static IP configuration.

Special feature: To control the default PXE behavior for an entire subnet, this field can also be passed in using CIDR notation. If `--ip-address` is CIDR, do not specify any other arguments other than `--name` and `--profile`.

When using the CIDR notation trick, don't specify any arguments other than `--name` and `--profile`... they won't be used.

–ipv6-address

The IPv6 address to use for this interface.

Note: This is not mutually exclusive with the `--ipv6-autoconfiguration` option, as interfaces can have many IPv6 addresses.

–ipv6-autoconfiguration

Use autoconfiguration mode to obtain the IPv6 address for this interface.

–ipv6-default-device

The default IPv6 device.

–ipv6-secondaries

The list of IPv6 secondaries for this interface.

–ipv6-mtu

Same as `--mtu`, however specific to the IPv6 stack for this interface.

–ipv6-static-routes

Same as `--static-routes`, however specific to the IPv6 stack for this interface.

–ipv6-default-gateway

This is the default gateway to use for this interface, specific only to the IPv6 stack. Unlike `--gateway`, this is set per-interface.

–mac-address (–mac)

Specifying a mac address via `--mac` allows the system object to boot directly to a specific profile via PXE, bypassing cobbler's PXE menu. If the name of the cobbler system already looks like a mac address, this is inferred from the system name and does not need to be specified.

MAC addresses have the format `AA:BB:CC:DD:EE:FF`. It's higly recommended to register your MAC-addresses in Cobbler if you're using static adressing with multiple interfaces, or if you are using any of the advanced networking features like bonding, bridges or VLANs.

Cobbler does contain a feature (enabled in `/etc/cobbler/settings`) that can automatically add new system records when it finds profiles being provisioned on hardware it has seen before. This may help if you do not have a report of all the MAC addresses in your datacenter/lab configuration.

–mtu

Sets the MTU (max transfer unit) property for the interface. Normally, this is set to 9000 to enable jumbo frames, but remember you must also enable it on in your switch configuration to function properly.

–management

When set to true, this interface will take precedence over others as the communication link to the Cobbler server. This means it will be used as the default kickstart interface if there are multiple interfaces to choose from.

–static

Indicates that this interface is statically configured. Many fields (such as gateway/subnet) will not be used unless this field is enabled. When Cobbler is managing DHCP, this will result in a static lease entry being created in the `dhcpd.conf`.

–static-routes

This is a space delimited list of ip/mask:gateway routing information in that format, which will be added as extra routes on the system. Most systems will not need this information.

```
--static-routes="192.168.1.0/16:192.168.1.1 172.16.0.0/16:172.16.0.1"
```

–netmask (formerly –subnet)

This is the netmask of the interface, for example 255.255.255.0.

–virt-bridge

(Virt-only) When specified, koan will associate the given interface with the physical bridge on the system. If no bridge is specified, this value will be inherited from the profile, which in turn may be inherited from the default virt bridge configured in *Cobbler Settings*.

Get Kickstart (getks)

The getks command shows the rendered kickstart/response file (preseed, etc.) for the given system. This is useful for previewing what will be downloaded from Cobbler when the system is building. This is also a good opportunity to catch snippets that are not rendering correctly.

As with remove, the `--name` option is required and is the only valid argument.

Example:

```
$ cobbler system getks --name=foo | less
```

Power Commands

By configuring the `--power-*` options above, Cobbler can be used to power on/off and reboot systems in your environment.

Example:

```
$ cobbler system poweron --name=foo
```

Please see the *Power Management* section for more details on using these commands.

Images

Cobbler can help with booting images physically and virtually, though the usage of these commands varies substantially by the type of image. Non-image based deployments are generally easier to work with and lead to more sustaintable infrastructure.

Repos

Repository mirroring allows cobbler to mirror not only install trees (`cobbler import` does this for you) but also optional packages, 3rd party content, and even updates. Mirroring all of this content locally on your network will result in faster, more up-to-date installations and faster updates. If you are only provisioning a home setup, this will probably be overkill, though it can be very useful for larger setups (labs, datacenters, etc). For information on how to keep your mirror up-to-date, see *Reposync*.

Example:

```
$ cobbler repo add --mirror=url --name=string [--rpmlist=list] [--creatrepo-
↪flags=string] \
[--keep-updated=Y/N] [--priority=number] [--arch=string] [--mirror-locally=Y/
↪N] [--breed=yum|rsync|rhn]
```

mirror

The addresss of the yum mirror. This can be an rsync:// URL, an ssh location, or a http:// or ftp:// mirror location. Filesystem paths also work.

The mirror address should specify an exact repository to mirror – just one architecture and just one distribution. If you have a seperate repo to mirror for a different arch, add that repo seperately.

Example:

```
rsync://yourmirror.example.com/fedora-linux-core/updates/6/i386 (for rsync␣
↪protocol)
http://mirrors.kernel.org/fedora/extras/6/i386/ (for http://)
user@yourmirror.example.com/fedora-linux-core/updates/6/i386  (for SSH)
```

Experimental support is also provided for mirroring RHN content when you need a fast local mirror. The mirror syntax for this is `--mirror=rhn://channel-name` and you must have entitlements for this to work. This requires the cobbler server to be installed on RHEL5 or later. You will also need a version of yum-utils equal or greater to 1.0.4.

name

This name is used as the save location for the mirror. If the mirror represented, say, Fedora Core 6 i386 updates, a good name would be "fc6i386updates". Again, be specific.

This name corresponds with values given to the `--repos` parameter of `cobbler profile add`. If a profile has a `--repos` value that matches the name given here, that repo can be automatically set up during provisioning (when supported) and installed systems will also use the boot server as a mirror (unless `yum_post_install_mirror` is disabled in the settings file). By default the provisioning server will act as a mirror to systems it installs, which may not be desirable for laptop configurations, etc.

Distros that can make use of yum repositories during kickstart include FC6 and later, RHEL 5 and later, and derivative distributions.

See the documentation on `cobbler profile add` for more information.

rpm-list

By specifying a space-delimited list of package names for `--rpm-list`, one can decide to mirror only a part of a repo (the list of packages given, plus dependencies). This may be helpful in conserving time/space/bandwidth. For instance, when mirroring FC6 Extras, it may be desired to mirror just cobbler and koan, and skip all of the game packages. To do this, use `--rpm-list="cobbler koan"`.

This option only works for `http://` and `ftp://` repositories (as it is powered by yumdownloader). It will be ignored for other mirror types, such as local paths and `rsync://` mirrors.

createrepo-flags

Specifies optional flags to feed into the createrepo tool, which is called when `cobbler reposync` is run for the given repository. The defaults are `-c cache`.

keep-updated

Specifies that the named repository should not be updated during a normal `cobbler reposync`. The repo may still be updated by name. The repo should be synced at least once before disabling this feature See `cobbler reposync` below.

mirror-locally

When set to "N", specifies that this yum repo is to be referenced directly via kickstarts and not mirrored locally on the cobbler server. Only `http://` and `ftp://` mirror urls are supported when using `--mirror-locally=N`, you cannot use filesystem URLs.

priority

Specifies the priority of the repository (the lower the number, the higher the priority), which applies to installed machines using the repositories that also have the yum priorities plugin installed. The default priority for the plugin is 99, as is that of all cobbler mirrored repositories.

arch

Specifies what architecture the repository should use. By default the current system arch (of the server) is used, which may not be desirable. Using this to override the default arch allows mirroring of source repositories (using `--arch=src`).

yumopts

Sets values for additional yum options that the repo should use on installed systems. For instance if a yum plugin takes a certain parameter "alpha" and "beta", use something like `--yumopts="alpha=2 beta=3"`.

breed

Ordinarily cobbler's repo system will understand what you mean without supplying this parameter, though you can set it explicitly if needed.

Management Classes

Management classes allow cobbler to function as a configuration management system. The lego blocks of configuration management, resources are grouped together via Management Classes and linked to a system. Cobbler supports two (2) resource types, which are configured in the order listed below:

1. *Package Resources*
2. *File Resources*

To add a Management Class, you would run the following command:

```
$ cobbler mgmtclass add --name=string --comment=string [--packages=list] [--
↪files=list]
```

name

The name of the mgmtclass. Use this name when adding a management class to a system, profile, or distro. To add a mgmtclass to an existing system use something like (`cobbler system edit --name="madhatter" --mgmt-classes="http mysql"`).

comment

A comment that describes the functions of the management class.

packages

Specifies a list of package resources required by the management class.

files

Specifies a list of file resources required by the management class.

File Resources

File resources are managed using cobbler file add, allowing you to create and delete files on a system.

Actions

create

Create the file. [Default]

remove

Remove the file.

Attributes

mode

Permission mode (as in chmod).

group

The group owner of the file.

user

The user for the file.

path

The path for the file.

template

The template for the file.

Example:

```
$ cobbler file add --name=string --comment=string [--action=string] --
↪mode=string --group=string \
--user=string --path=string [--template=string]
```

Package Resources

Package resources are managed using cobbler package add, allowing you to install and uninstall packages on a system outside of your install process.

Actions

install

Install the package. [Default]

uninstall

Uninstall the package.

Attributes

installer

Which package manager to use, vaild options [rpm|yum].

version

Which version of the package to install.

Example:

```
$ cobbler package add --name=string --comment=string [--
↪action=install|uninstall] --installer=string \
[--version=string]
```

3.2 Cobbler Direct Commands

3.2.1 Check

The check command is used to provide information to the user regarding possible issues with their installation. Many of these checks are feature-based, and may not show up depending on the features you have enabled in Cobbler.

One of the more important things to remember about the check command is that the output contains suggestions, and not absolutes. That is, the check output may always show up (for example, the SELinux check when it is enabled on the system), or the suggested remedy is not required to make Cobbler function properly (for example, the firewall checks). It is very important to evaluate each item in the listed output individually, and not be concerned with them unless you are having definite problems with functionality.

Example:

```
$ cobbler check
The following are potential configuration items that you may want to fix:

1 : SELinux is enabled. Please review the following wiki page for details on␣
↪ensuring cobbler works correctly in your SELinux environment:
    https://github.com/cobbler/cobbler/wiki/Selinux
2 : comment 'dists' on /etc/debmirror.conf for proper debian support
3 : comment 'arches' on /etc/debmirror.conf for proper debian support
4 : Dynamic settings changes are enabled, be sure you run "sed -i 's/^
↪[[:space:]]\+/ /' /etc/cobbler/settings" to ensure the settings file is␣
↪properly indented

Restart cobblerd and then run 'cobbler sync' to apply changes.
```

3.2.2 Sync

The sync command is very important, though very often unnecessary for most situations. It's primary purpose is to force a rewrite of all configuration files, distribution files in the TFTP root, and to restart managed services. So why is it unnecessary? Because in most common situations (after an object is edited, for example), Cobbler executes what is known as a "lite sync" which rewrites most critical files.

When is a full sync required? When you are using `manage_dhcpd` *DHCP* with systems that use static leases. In that case, a full sync is required to rewrite the `dhcpd.conf` file and to restart the dhcpd service. Adding support for OMAPI is on the roadmap, which will hopefully relegate full syncs to troubleshooting situations.

Example: `$ cobbler sync`

How Sync Works

A full sync will perform the following actions:

1. Run pre-sync *Triggers*
2. Clean the TFTP tree of any and all files
3. Re-copy boot loaders to the TFTP tree
4. **Re-copy distribution files to the TFTP tree**
    - This will attempt to hardlink files if possible
5. Rewrite the pxelinux.cfg/default file
6. Rewrite all other pxelinux.cfg files for systems
7. Rewrite all managed config files (DHCP, DNS, etc.) and restarts services
8. Cleans the "link cache"
9. Executes post-sync and change *Triggers*

As noted above, this can take quite a bit of time if there are many distributions.

See also: *Managing Services with Cobbler*

3.2.3 Distro Signatures

Prior to Cobbler 2.4.0, import modules for each supported distro were separate and customized for each specific distribution. The values for breed and os-version were hard-coded into cobbler, so adding support for new distros or newer versions of an already supported distro required code changes and a complete Cobbler upgrade.

Cobbler 2.4.0 introduces the concept of distro signatures to make adding support for newer distro versions without requiring an upgrade to the rest of the system.

Distro Signatures File

The distro signatures are stored in `/var/lib/cobbler/distro_signatures.json`. As the extension indicates, this is a JSON-formatted file, with the following structure:

```
{
    "breeds": {
        "<breed-name>": {
            "<os-version1>": {
                "signatures": "...",
                "default_kickstart":"..."
            }
        },
        "<breed-name>": {
            "<os-version1>": {
                "signatures": "...",
                "default_kickstart":"...",
            }
        }
    }
}
```

This file is read in when cobblerd starts, and logs a message noting how many breeds and os-versions it has loaded:

```
INFO | 9 breeds and 21 OS versions read from the signature file
```

3.2.4 CLI Commands

The signature CLI command has the following sub-commands:

```
$ cobbler signature --help
usage
=====
cobbler signature report
cobbler signature update
```

cobbler signature report

This command prints out a report of the currently loaded signatures and os-versions.

```
$ cobbler signature report
Currently loaded signatures:
debian:
    squeeze
freebsd:
    8.2
    8.3
    9.0
```

(continues on next page)

(continued from previous page)

```
generic:
    (none)
redhat:
    fedora16
    fedora17
    fedora18
    rhel4
    rhel5
    rhel6
suse:
    opensuse11.2
    opensuse11.3
    opensuse11.4
    opensuse12.1
    opensuse12.2
ubuntu:
    oneiric
    precise
    quantal
unix:
    (none)
vmware:
    esx4
    esxi4
    esxi5
windows:
    (none)

9 breeds with 21 total signatures loaded
```

An optional `--name` parameter can be specified to limit the report to one breed:

```
$ cobbler signature report --name=ubuntu
Currently loaded signatures:
ubuntu:
    oneiric
    precise
    quantal

Breed 'ubuntu' has 3 total signatures
```

cobbler signature update

This command will cause Cobbler to go and fetch the latest distro signature file from http://cobbler.github.con/signatures/latest.json, and load the signatures in that file. This file will be tested first, to ensure it is formatted correctly.

```
cobbler signature update
task started: 2012-11-21_222926_sigupdate
task started (id=Updating Signatures, time=Wed Nov 21 22:29:26 2012)
```

(continues on next page)

(continued from previous page)

```
Successfully got file from http://cobbler.github.com/signatures/latest.json
*** TASK COMPLETE ***
```

This command currently takes no options.

3.2.5 Import

The purpose of `cobbler import` is to set up a network install server for one or more distributions. This mirrors content based on a DVD image, an ISO file, a tree on a mounted filesystem, an external rsync mirror or SSH location.

```
$ cobbler import --path=/path/to/distro --name=F12
```

This example shows the two required arguments for import: `--path` and `--name`.

Alternative set-up from existing filesystem

(Note: the description of "–available-as" is probably inadequate.)

What if you don't want to mirror the install content on your install server? Say you already have the trees from all your DVDs and/or CDs extracted on a Filer mounted over NFS somewhere. This works too, with the addition of one more argument:

```
cobbler import --path=/path/where/filer/is/mounted --name=filer \
  --available-as=nfs://nfsserver.example.org:/is/mounted/here
```

The above command will set up cobbler automatically using all of the above distros (stored on the remote filer) – but will keep the trees on NFS. This saves disk space on the Cobbler server. As you add more distros over time to the filer, you can keep running the above commands to add them to Cobbler.

Importing Trees

(**Note:** this topic was imported from "Advanced Topics", and needs to be more properly integrated into this document.)

(**Note:** the description of "–available-as" is probably inadequate.)

Cobbler can auto-add distributions and profiles from remote sources, whether this is a filesystem path or an rsync mirror. This can save a lot of time when setting up a new provisioning environment. Import is a feature that many users will want to take advantage of, and is very simple to use.

After an import is run, cobbler will try to detect the distribution type and automatically assign kickstarts. By default, it will provision the system by erasing the hard drive, setting up eth0 for dhcp, and using a default password of "cobbler". If this is undesirable, edit the kickstart files in `/var/lib/cobbler/kickstarts` to do something else or change the kickstart setting after cobbler creates the profile.

Mirrored content is saved automatically in `/var/www/cobbler/ks_mirror`.

- Example 1: `cobbler import --path=rsync://mirrorserver.example.com/path/ --name=fedora --arch=x86`
- Example 2: `cobbler import --path=root@192.168.1.10:/stuff --name=bar`
- Example 3: `cobbler import --path=/mnt/dvd --name=baz --arch=x86_64`
- Example 4: `cobbler import --path=/path/to/stuff --name=glorp`
- Example 5: `cobbler import --path=/path/where/filer/is/mounted --name=anyname --available-as=nfs://nfs.example.org:/where/mounted/`

Once imported, run a `cobbler list` or `cobbler report` to see what you've added.

By default, the rsync operations will exclude content of certain architectures, debug RPMs, and ISO images – to change what is excluded during an import, see `/etc/cobbler/rsync.exclude`.

Note that all of the import commands will mirror install tree content into `/var/www/cobbler` unless a network accessible location is given with `--available-as`. `--available-as` will be primarily used when importing distros stored on an external NAS box, or potentially on another partition on the same machine that is already accessible via `http://` or `ftp://`.

For import methods using rsync, additional flags can be passed to rsync with the option `--rsync-flags`.

Should you want to force the usage of a specific cobbler kickstart template for all profiles created by an import, you can feed the option `--kickstart` to import, to bypass the built-in kickstart auto-detection.

Kickstarts

Kickstarts are answer files that script the installation of the OS. Well, for Fedora and Red Hat based distributions it is called kickstart. We also support other distributions that have similar answer files, but let's just use kickstart as an example for now. The kickstarts automatically assigned above will install physical machines (or virtual machines – we'll get to that later) with a default password of "cobbler" (don't worry, you can change these defaults) and a really basic set of packages. For something more complicated, you may wish to edit the default kickstarts in */var/lib/cobbler/kickstarts*. You could also use cobbler to assign them new kickstart files. These files are actually [Kickstart Templates](Kickstart Templating), a level beyond regular kickstarts that can make advanced customizations easier to achieve. We'll talk more about that later as well.

Associated server set-up

Firewall

Depending on your usage, you will probably need to make sure iptables is configured to allow access to the right services. Here's an example configuration:

```
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
```

(continues on next page)

(continued from previous page)

```
-A INPUT -p icmp --icmp-type any -j ACCEPT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# LOCALHOST
-A INPUT -i lo -j ACCEPT

# SSH
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
# DNS - TCP/UDP
-A INPUT -m state --state NEW -m udp -p udp --dport 53 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 53 -j ACCEPT
# DHCP
-A INPUT -m state --state NEW -m udp -p udp --dport 68 -j ACCEPT
# TFTP - TCP/UDP
-A INPUT -m state --state NEW -m tcp -p tcp --dport 69 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 69 -j ACCEPT
# NTP
-A INPUT -m state --state NEW -m udp -p udp --dport 123 -j ACCEPT
# HTTP/HTTPS
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT
# Syslog for cobbler
-A INPUT -m state --state NEW -m udp -p udp --dport 25150 -j ACCEPT
# Koan XMLRPC ports
-A INPUT -m state --state NEW -m tcp -p tcp --dport 25151 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 25152 -j ACCEPT

#-A INPUT -j LOG
-A INPUT -j REJECT --reject-with icmp-host-prohibited

COMMIT
```

Adapt this to your own environment.

SELinux

Most likely you are using SELinux since it has been in the Linux mainline since 2.6, as a result you'll need to allow network access from the Apache web server.

```
setsebool -P httpd_can_network_connect true
```

Services

Depending on whether you are running DHCP and DNS on the same box, you will want to enable various services:

```
/sbin/service httpd start
/sbin/service dhcpd start
/sbin/service xinetd start
/sbin/service cobblerd start

/sbin/chkconfig httpd on
/sbin/chkconfig dhcpd on
/sbin/chkconfig xinetd on
/sbin/chkconfig tftp on
/sbin/chkconfig cobblerd on
```

This command `cobbler check` should inform you of most of this.

Using the server

PXE

PXE for network installation of "bare metal" machines is straightforward. You need to set up DHCP:

- If the DHCP server is somewhere else, not on the Cobbler server, its administrator should set its `next-server` to specify your cobbler server.
- If you run DHCP locally and want Cobbler manage it for you, set `manage_dhcp` to 1 in `/etc/cobbler/settings`, edit `/etc/cobbler/dhcp.template` to change some defaults, and re-run `cobbler sync`. See *DHCP* for further details.

Once you get PXE set up, all of the bare-metal compatible profiles will, by name, show up in PXE menus when the machines network boot. Type "menu" at the prompt and choose one from the list. Or just don't do anything and the machine will default through to local booting. (Some Xen paravirt profiles will not show up, because you cannot install these on physical machines – this is intended)

Should you want to pin a particular system to install a particular profile the next time it reboots, just run:

```
cobbler system add --name=example --mac=$mac-address --profile=$profile-name
```

Then the above machine will boot directly to the profile of choice without bringing up the menu. Don't forget to read the manpage docs as there are more options for customization and control available. There are also lots of useful settings described in `/etc/cobbler/settings` that you will want to read over.

Reinstallation

Should you have a system you want to install that Fedora 12 on (instead of whatever it is running now), right now, you can do this:

```
yum install koan
koan --server=bootserver.example.com --list=profiles
koan --replace-self --server=bootserver.example.com --profile=F12-i386
/sbin/reboot
```

The system will install the new operating system after rebooting, hands off, no interaction required.

Notice in the above example "F12-i386" is just one of the boring default profiles cobbler created for you. You can also create your own, for instance "F12-webservers" or "F12-appserver" – whatever you would like to automate.

Virtualization

Want to install a virtual guest instead (perhaps Xen or KVM)? No problem.

```
yum install koan
koan --server=bootserver.example.com --virt --virt-type=xenpv --profile=F12-
↪i386-xen
```

Done.

You can also use KVM or other virtualization methods. These are covered elsewhere on the Wiki. Some distributions have Xen specific profiles you need to use, though this is merged back together starting with Fedora 12.

3.2.6 Reposync

Yum repository management is an optional feature, and is not required to provision through cobbler. However, if cobbler is configured to mirror certain repositories, it can then be used to associate profiles with those repositories. Systems installed under those profiles will then be autoconfigured to use these repository mirrors in `/etc/yum.repos.d`, and if supported (Fedora Core 6 and later) these repositories can be leveraged even within Anaconda. This can be useful if (A) you have a large install base, (B) you want fast installation and upgrades for your systems, or (C) have some extra software not in a standard repository but want provisioned systems to know about that repository.

Make sure there is plenty of space in cobbler's webdir, which defaults to `/var/www/cobbler`.

```
$ cobbler reposync [--tries=N] [--no-fail]
```

Cobbler reposync is the command to use to update repos as configured with `cobbler repo add`. Mirroring can take a long time, and usage of cobbler reposync prior to usage is needed to ensure provisioned systems have the files they need to actually use the mirrored repositories. If you just add repos and never run `cobbler reposync`, the repos will never be mirrored. This is probably a command you would want to put on a crontab, though the frequency of that crontab and where the output goes is left up to the systems administrator.

For those familiar with yum's reposync, cobbler's reposync is (in most uses) a wrapper around the yum command. Please use `cobbler reposync` to update cobbler mirrors, as yum's reposync does not perform all required steps. Also cobbler adds support for rsync and SSH locations, where as yum's reposync only supports what yum supports (http/ftp).

If you ever want to update a certain repository you can run:

```
$ cobbler reposync --only="reponame1" ...
```

When updating repos by name, a repo will be updated even if it is set to be not updated during a regular reposync operation (ex: `cobbler repo edit --name=reponame1 --keep-updated=0`).

Note that if a cobbler import provides enough information to use the boot server as a yum mirror for core packages, cobbler can set up kickstarts to use the cobbler server as a mirror instead of the outside world. If this feature is desirable, it can be turned on by setting `yum_post_install_mirror` to `1` in `/etc/cobbler/settings` (and running `cobbler sync`). You should not use this feature if machines are provisioned on a different VLAN/network than production, or if you are provisioning laptops that will want to acquire updates on multiple networks.

The flags `--tries=N` (for example, `--tries=3`) and `--no-fail` should likely be used when putting reposync on a crontab. They ensure network glitches in one repo can be retried and also that a failure to synchronize one repo does not stop other repositories from being synchronized.

3.2.7 Build ISO

Often an environment cannot support PXE because of either (A) an unfortunate lack of control over DHCP configurations (i.e. another group owns DHCP and won't give you a next-server entry), or (B) you are using static IPs only.

This is easily solved: `cobbler buildiso`

What this command does is to copy all distro kernel/initrds onto a boot CD image and generate a menu for the ISO that is essentially equivalent to the PXE menu provided to net-installing machines via Cobbler.

By default, the boot CD menu will include all profiles and systems, you can force it to display a list of profiles/systems in concern with the following.

Cobbler versions >= 2.2.0:

```
# cobbler buildiso --systems="system1 system2 system3"
# cobbler buildiso --profiles="profile1 profile2 profile3"
```

Cobbler versions < 2.2.0:

```
# cobbler buildiso --systems="system1,system2,system3"
# cobbler buildiso --profiles="profile1,profile2,profile3"
```

If you need to install into a lab (or other environment) that does not have network access to the cobbler server, you can also copy a full distribution tree plus profile and system records onto a disk image:

```
# cobbler buildiso --standalone --distro="distro1"
```

3.2.8 Command Line Search

line search can be used to ask questions about your cobbler configuration, rather than just having to run `cobbler list` or `cobbler report` and scanning through the results. (The *Web Interface* also supports search/filtering, for those that want to use it, though that is not documented on this page)

Command line search works on all distro/profile/system/and repo objects.

```
cobbler distro find --help
cobbler profile find --help
cobbler system find --help
cobbler repo find --help
```

Note: Some of these examples are kind of arbitrary. I'm sure you can think of some more real world examples.

Examples

Find what system record has a given mac address.

```
cobbler system find --mac=AA:BB:CC:DD:EE:FF
```

If anything is using a certain kernel, delete that object and all it's children (profiles, systems, etc).

```
cobbler distro find --kernel=/path/to/kernel | xargs -n1 --replace cobbler␣
↪distro remove --name={} --recursive
```

What profiles use the repo "epel-5-i386-testing" ?

```
cobbler profile find --repos=epel-5-i386-testing
```

Which profiles are owned by neo AND mranderson?

```
cobbler profile find --owners="neo,mranderson"
# lists need to be comma delimited, like this, with no unneeded spaces
```

What systems are set to pass the kernel argument "color=red" ?

```
cobbler system find --kopts="color=red"
```

What systems are set to pass the kernel argument "color=red" and "number=5" ?

```
cobbler system find --kopts="color=red number=5"
# space delimited key value pairs
# key1=value1 key2 key3=value3
```

What systems set the kickstart metadata variable of foo to the value 'bar' ?

```
cobbler system find --ksmeta="foo=bar"
# space delimited key value pairs again
```

What systems are set to netboot disabled?

```
cobbler system find --netboot-enabled=0
# note, this also accepts 'false', or 'no'
```

For all systems that are assigned to profile "foo" that are set to netboot disabled, enable them.

```
cobbler system find --profile=foo --netboot-enabled=0 | xargs -n1 --replace␣
↪cobbler system edit --name={} --netboot-enabled=1
# demonstrates an "AND" query combined with xargs usage.
```

A Note About Types And Wildcards

Though the cobbler objects behind the scenes store data in various different formats (booleans, hashes, lists, strings), it all works fom the command line as text.

If multiple terms are specified to one argument, the search is an "AND" search.

If multiple parameters are specified, the search is still an "AND" search.

The find command understands patterns such as "*" and "?". This is supported using Python's fnmatch.

To learn more: `pydoc fnmatch.fnmatch`

All systems starting with the string foo: `cobbler system find --name="foo*"`

This is rather useful when combined with xargs. This is a rather tame example, reporting on all systems starting with "TEST".

```
cobbler system find --name="TEST*" | xargs -n1 --replace cobbler system␣
↪report --name={}
```

By extension, you could use this to toggle the `--netboot-enabled` systems of machines with certain hostnames, mac addresses, and so forth, or perform other kinds of wholesale edits (for instance, deletes, or assigning profiles with certain names to new distros when upgrading them from F8 to F9, for instance).

API Usage

All of this functionality is also exposed through the API

```
#!/usr/bin/python
import cobbler.api as capi
api_handle = capi.BootAPI()
matches = api_handle.find_profile(name="TEST*",return_list=True)
print matches
```

You will find uses of `.find()` throughout the cobbler code that make use of this behavior.

3.2.9 Replication

```
cobbler replicate --help
```

Replication works by downloading the configuration from one cobbler server into another. It is useful for Highly Available setups, disaster recovery, support of multiple geographies, or for load balancing.

```
cobbler replicate --master=master.example.org
```

With the default arguments, only distribution and profile metadata are synchronized. Without any of the other sync flags (described below) it is assumed data backing these objects (such as kernels/initrds, etc) are already accessible. Don't worry though, cobbler can help move those over too.

Transferring More Than Just Metadata

Cobbler can transfer mirrored trees, packages, snippets, kickstart templates, and triggers as well. To do this, just use the appropriate flags with cobbler replicate.

```
[root@localhost mdehaan]# cobbler replicate --help
Usage: cobbler [options]

Options:
  -h, --help            show this help message and exit
  --master=MASTER       Cobbler server to replicate from.
  --distros=PATTERN     pattern of distros  to replicate
  --profiles=PATTERN    pattern of profiles to replicate
  --systems=PATTERN     pattern of systems to replicate
  --repos=PATTERN       pattern of repos to replicate
  --image=PATTERN       pattern of images to replicate
  --omit-data           do not rsync data
  --prune               remove objects (of all types) not found on the master
```

Setup

On each replica-to-be cobbler server, just install cobbler as normal, and make sure `/etc/cobbler/settings` and `/etc/cobbler/modules.conf` are appropriate. Use `cobbler check` to spot check your work. Cobbler replicate will not configure these files, and you may want different site-specific settings for variables in these files. That's fine, as cobbler replicate will respect these.

How It Works

Metadata is transferred over Cobbler XMLRPC, so you'll need to have the Cobbler XMLRPC endpoint accessible – `http://servername:80/cobbler\_api`. This is the read only API so no authentication is required. This is possible because this is a user-initiated pull operation, not a push operation.

Files are transferred either by rsync (over ssh) or scp, so you will probably want to use ssh-agent prior to kicking off the replicate command, or otherwise use authorized_keys on the remote host to save typing.

Limitations

It is perfectly fine to sync data bi-directionally, though keep in mind metadata being synced is not timestamped with the time of the last edit (this may be a nice future extension), so the latest sync "wins". Cobbler replicate is, generally, designed to have a "master" concept, so it is probably not desirable yet to do bi-directional syncing.

Common Use Cases

High Availability / Disaster Recovery

A remote cobbler server periodically replicates from the master to keep an active installation.

Load Balancing

Similar to the HA/Disaster Recovery case, consider using a *Triggers* to notify the other server to pull new metadata when commands are issued.

Multiple Geographies

Several remote servers pull from the master, either triggered by a *Triggers* on the central server, or otherwise on daily cron. This allows for establishing install mirrors that are closer and therefore faster and less bandwidth hungry. The admin can choose whether or not system records should be centrally managed. It may be desirable to just centrally provide the distributions and profiles and keep the system records on each seperate cobbler server, however, there is nothing to say all records can't be kept centrally as well. (Choose one or the other, don't do a mixture of both.)

3.2.10 Validate Kickstart

3.2.11 ACL Setup

Cobbler contains an "aclsetup" command for automation of setting up file system acls (i.e. setfacl) on directories that cobbler needs to read and write to.

Using File System ACLs

Usage of this command allows the administrator to grant access to other users without granting them the ability to run cobbler as root.

```
$ cobbler aclsetup --help
Usage: cobbler aclsetup  [ARGS]

Options:
  -h, --help            show this help message and exit
  --adduser=ADDUSER     give acls to this user
  --addgroup=ADDGROUP   give acls to this group
  --removeuser=REMOVEUSER
                        remove acls from this user
  --removegroup=REMOVEGROUP
                        remove acls from this group
```

Example:

```
$ cobbler aclsetup --adduser=timmy
```

The above example gives timmy access to run cobbler commands.

Note that aclsetup does grant access to configure all of `/etc/cobbler`, `/var/www/cobbler`, and `/var/lib/cobbler`, so it is still rather powerful in terms of the access it grants (though somewhat less so than providing root).

A user with acls can, for instance, edit cobbler triggers which are later run by cobblerd (as root). In this event, cobbler access (either sudo or aclsetup) should not be granted to users you do not trust completely. This should not be a major problem as in giving them access to configure future aspects of your network (via the provisioning server) they are already being granted fairly broad rights.

It is at least nicer than running "sudo" all of the time if you were going to grant a user "no password" sudo access to cobbler.

3.2.12 Dynamic Settings

The CLI command for dynamic settings has two sub-commands:

```
$ cobbler setting --help
usage
=====
cobbler setting edit
cobbler setting report
```

cobbler setting edit

This command allows you to modify a setting on the fly. It takes affect immediately, however depending on the setting you change, a `cobbler sync` may be required afterwards in order for the change to be fully applied.

This syntax of this command is as follows:

```
$ cobbler setting edit --name=option --value=value
```

As with other cobbler primitives, settings that are array-based should be space-separated while hashes should be a space-separated list of `key=value` pairs.

cobbler setting report

This command prints a report of the current settings. The syntax of this command is as follows:

```
$ cobbler setting report [--name=option]
```

The list of settings can be limited to a single setting by specifying the –name option.

3.2.13 Version

The Cobbler version command is very simple, and provides a little more detailed information about your installation.

Example:

```
$ cobbler version
Cobbler 2.4.0
  source: ?, ?
  build time: Sun Nov 25 11:45:24 2012
```

The first piece of information is the version. The second line includes information regarding the associated commit for this version. In official releases, this should correspond to the commit for which the build was tagged in git. The final line is the build time, which could be the time the RPM was built, or when the "make" command was run when installing from source.

All of this information is useful when asking for help, so be sure to provide it when opening trouble tickets.

3.3 Cobbler Settings

Cobbler has many features, many of which are disabled out of the box for simplicity's sake. Various settings are used to enable these features and to modify how they work, while other settings are used to provide default functionality to base features (for example, the default encrypted password to use).

3.3.1 Dynamic Settings

Prior to Cobbler 2.4.0, any changes to `/etc/cobbler/settings` required a restart of the cobblerd daemon for those changes to take affect. Now, with 2.4.0+, you can easily modify settings on the fly via the `cobbler setting` command.

Enabling Dynamic Settings

Dynamic settings are not enabled by default. In order to enable them, you must set "allow_dynamic_settings: 1" in `/etc/cobbler/settings` and restart cobblerd.

Caveats

Over the years, the Cobbler settings file has grown organically, and as such has not always had consistent spacing applied to the YAML entries it contains. In order to ensure that augeas can correctly rewrite the settings, you must run the following sed command:

```
$ sed -i 's/^[[:space:]]\+/ /' /etc/cobbler/settings
```

When dynamic settings are enabled, the `cobbler check` command will also print out this recommendation.

CLI Commands

Please see the *Dynamic Settings CLI Command* section for details on the dynamic settings commands.

3.3.2 Complete Settings List

This page documents all settings `/etc/cobbler/settings` available for configuring both cobblerd and the cobbler CLI command. Be sure to restart the cobblerd service after making changes to this file.

Note: The defaults shown here are noted via JSON syntax. The settings file is stored as YAML, so be sure to format it correctly or cobblerd and the CLI command will not work properly.

---

allow_duplicate_hostnames

- **type:** Boolean
- **default:** 0
- **Description:** If set, Cobbler will allow multiple systems to use the same FQDN for the `--dns-name` interface option. This field is used for system identification for things like configuration management integration, so take care when enabling it.

allow_duplicate_ips

- **type:** Boolean
- **default:** 0
- **Description:** If set, Cobbler will allow multiple systems to use the same IP address for interfaces. If enabled, this could impact managed services like DHCP and DNS where multiple active systems conflict.

allow_duplicate_macs

- **type:** Boolean
- **default:** 0
- **Description:** If set, Cobbler will allow multiple systems to use the same MAC address for interfaces. If enabled, this could impact managed services like DHCP and DNS where multiple active systems conflict.

allow_dynamic_settings

- **type:** Boolean

- **default:** 0
- **Description:** If enabled, Cobbler will allow settings to be modified on the fly without a restart to the cobblerd daemon. Please reference the *Dynamic Settings* section for more details.

anamon_enabled

- **type:** Boolean
- **default:** 0
- **Description:** If set, anamon will be enabled during the Anaconda kickstart process. This is specific to Red Hat style kickstarts only.

Please refer to the *Anaconda Monitoring* section for more details.

bind_chroot_path

- **type:** String
- **default:** ""
- **Description:** This sets the path of the directory in which bind-chroot compatible configuration files will be created. In most situations, this should be automatically detected by default (set to an empty string).

Please refer to the *DNS* section for more details.

bind_master

- **type:** String
- **default:** "127.0.0.1"
- **Description:** The bind master to use when creating slave DNS zones.

Please refer to the *DNS* section for more details.

build_reporting_email

- **type:** Array of Strings
- **default:** ['root@localhost ('root@localhost)']
- **Description:** A list of email addresses to send build reports to.

build_reporting_enabled

- **type:** Boolean
- **default:** 0

- **Description:** Setting this option enables build reporting emails.

build_reporting_sender

- **type:** String
- **default:** ""
- **Description:** The email address to use as the sender of a build report email (optional).

build_reporting_smtp_server

- **type:** String
- **default:** "localhost"
- **Description:** The SMTP server to use for build report emails.

build_reporting_subject

- **type:** String
- **default:** ""
- **Description:** This setting allows you to override the default auto-generated subject lines for build report emails.

build_reporting_to_address

- **type:** String
- **default:** ""
- **Description:** Not currently used.

buildisodir

- **type:** String
- **default:** "/var/cache/cobbler/buildiso"
- **Description:** The default directory to use as scratch space when building an ISO via Cobbler. This can be overridden on the command line.

Please refer to the *Build ISO* section for more details.

cheetah_import_whitelist

- **type:** Array of Strings
- **default:** ['random', 're', 'time']
- **Description:** This setting creates a whitelist of python modules that can be imported in a template.

This is a security issue, as allowing certain python modules would allow users to create templates that overwrite system files (ie. the os module) or execute shell commands (ie. the subprocess module). Make sure you understand the capabilities a python module has before adding them to this whitelist.

client_use_localhost

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, all commands will be forced to use the localhost address instead of the "server" setting. The cobbler client command can be used to manage remote cobblerd instances, so enabling this option would force all cobbler commands to operate locally only.

cobbler_master

- **type:** String
- **default:** ""
- **Description:** The default server to pull from when using the replicate command.

Please refer to the *Replication* section for more details.

consoles

- **type:** String
- **default:** "/var/consoles"
- **Description:** The path to the directory containing system consoles, used primarily for clearing logs and messages.

createrepo_flags

- **type:** String
- **default:** "-c cache -s sha –update"
- **Description:** Default options to use for the createrepo command when creating new repositories during a reposync.

If you have `createrepo >= 0.4.10`, consider `-c cache --update -C`, which can dramatically improve your `cobbler reposync` time. `-s sha` enables working with Fedora repos from F11/F12 from EL-4 or EL-5 without `python-hashlib` installed (which is not available on EL-4)

Please refer to the *Package Management and Mirroring* section for more details.

default_deployment_method

- **type:** String
- **default:** "ssh"
- **Description:** Not currently used.

default_kickstart

- **type:** String
- **default:** "/var/lib/cobbler/kickstarts/default.ks"
- **Description:** The default kickstart file to use if no other is specified. This option is effectively deprecated, as the default kickstart to use is now specified in the distro signatures configuration file. Please see the *Distro Signatures* section for more details.

default_name_servers

- **type:** Array of Strings
- **default:** []
- **Description:** A list of name servers to assign to all systems and profiles that are built. This will be used both pre and post install.

default_name_servers_search

- **type:** Array of Strings
- **default:** []
- **Description:** A list of domains to search by default. This will be inserted into the resolv.conf file.

default_ownership

- **type:** Array of Strings
- **default:** ['admin']
- **Description:** A list of owners to assign to newly created objects. This is used only for Web UI authorization.

Please refer to the *Web Authorization* section for more details.

default_password_crypted

- **type:** String
- **default:** "1wrWZXfa7$Ts7jMmpdZkTlu0lSx1A/I/" (cobbler)
- **Description:** The default hashed password to use in kickstarts. The default value is "cobbler" (hashed).

To generate a new hashed password, use the following command: `$ openssl passwd -1`

Be sure to enclose the hash with quotation marks.

default_template_type

- **type:** String
- **default:** "cheetah"
- **Description:** The default template type to use when parsing kickstarts and snippets. The default template type is Cheetah, and changing this value will currently break all snippets and templates currently shipped with Cobbler.

Please refer to the *Alternative template formats* section for more details.

default_virt_bridge

- **type:** String
- **default:** "xenbr0"
- **Description:** The default bridge to assign virtual interfaces to.

default_virt_disk_driver

- **type:** String
- **default:** "raw"
- **Description:** The default disk driver to use for virtual disks. Older versions of `python-virtinst` do not support changing this at build time, so this option will be ignored in those cases.

default_virt_file_size

- **type:** Integer
- **default:** 5
- **Description:** The default size (in gigabytes) to use for new virtual disks.

default_virt_ram

- **type:** Integer
- **default:** 512
- **Description:** The default size (in megabytes) of RAM to assign to new virtual machines.

default_virt_type

- **type:** String
- **default:** "xenpv"
- **Description:** The default virtualization type to use for virtual machines created with the koan utility.

Please refer to the https://koan.readthedocs.io/ section for more details.

enable_gpxe

- **type:** Boolean
- **default:** 0
- **Description:** If set, Cobbler will enable the use of gPXE.

Please refer to the *Using gPXE* section for more details.

enable_menu

- **type:** Boolean
- **default:** 1
- **Description:** If set, Cobbler will add each new profile entry to the default PXE boot menu. This can be overridden on a per-profile basis when adding/editing profiles with `--enable-menu=0/1`. Users should ordinarily leave this setting enabled unless they are concerned with accidental reinstalls from users who select an entry at the PXE boot menu. Adding a password to the boot menus templates may also be a good solution to prevent unwanted reinstallations.

func_auto_setup

- **type:** Boolean
- **default:** 0
- **Description:** If set, Cobbler will install and configure Func. This makes sure each installed machine is set up to use func out of the box, which is a powerful way to script and control remote machines.

Please refer to the *Func Integration* section for more details.

func_master

- **type:** String
- **default:** "overlord.example.org"
- **Description:** The Func master server (overlord) to use by default.

Please refer to the *Func Integration* section for more details.

http_port

- **type:** String
- **default:** "80"
- **Description:** The port on which Apache is listening. Only change this if your instance of Apache is listening on a different port (for example: 8080).

isc_set_host_name

- **type:** Boolean
- **default:** 0
- **Description:** Not currently used.

iso_template_dir

- **type:** String
- **default:** "/etc/cobbler/iso"
- **Description:** The directory containing the buildiso.template, which is a SYSLINUX style configuration file for use in the buildiso process.

Please refer to the *Build ISO* section for more details.

kerberos_realm

- **type:** String
- **default:** "EXAMPLE.COM"
- **Description:** Not currently used (all kerberos configuration must currently be done manually).

Please refer to the *Defer to Apache / Kerberos* section for more details.

kernel_options

- **type:** Dictionary
- **default:** {'ksdevice': 'bootif', 'lang': ' ', 'text': '~'}
- **Description:** A dictionary of key/value pairs that will be added to the kernel command line during the installation only (post-installation options are specified at the distro/profile/etc. object level).

By default, each key/value pair will be show up as key=value in the kernel command line. Setting the value for a given key to '~' (tilde) will cause the option to be printed by itself with no '='.

Note: The kernel command line has a maximum character limitation of 256 characters. Cobbler will print a warning if you exceed this limit.

---

kernel_options_s390x

- **type:** Dictionary
- **default:** {'vnc': '~', 'ip': False, 'RUNKS': 1, 'ramdisk_size': 40000, 'ro': '~', 'root': '/dev/ram0'}
- **Description:** Same as the kernel_options setting, but specific to s390x architectures.

ldap_anonymous_bind

- **type:** Boolean
- **default:** 1
- **Description:** If set, the LDAP authentication module will use an anonymous bind when connecting to the LDAP server.

Please refer to the *LDAP* section for more details.

ldap_base_dn

- **type:** String
- **default:** "DC=example,DC=com"
- **Description:** The base DN to use for LDAP authentication.

Please refer to the *LDAP* section for more details.

ldap_management_default_type

- **type:** String
- **default:** "authconfig"

- **Description:** Not currently used.

Please refer to the *LDAP* section for more details.

ldap_port

- **type:** Integer
- **default:** 389
- **Description:** The port to use when connecting to the LDAP server. If TLS is enabled and this port is the default of 389, cobbler will internally convert it to 636 for SSL.

Please refer to the *LDAP* section for more details.

ldap_search_bind_dn

- **type:** String
- **default:** ""
- **Description:** The DN to use for binding to the LDAP server for authentication, used only if `ldap_anonymous_bind=0`.

Please refer to the *LDAP* section for more details.

ldap_search_passwd

- **type:** String
- **default:** ""
- **Description:** The password to use when binding to the LDA server for authentication, used only if `ldap_anonymous_bind=0`.

Please refer to the *LDAP* section for more details.

ldap_search_prefix

- **type:** String
- **default:** "uid="
- **Description:** The prefix to use for searches when querying the LDAP server.

Please refer to the *LDAP* section for more details.

ldap_server

- **type:** Boolean
- **default:** "ldap.example.com"
- **Description:** The LDAP server to use for LDAP authentication.

Please refer to the *LDAP* section for more details.

ldap_tls

- **type:** Boolean
- **default:** 1
- **Description:** If set, the LDAP authentication will occur over a SSL/TLS encrypted connection.

Please refer to the *LDAP* section for more details.

ldap_tls_cacertfile

- **type:** Boolean
- **default:** 1
- **Description:** The CA certificate file to use when using TLS encryption.

Please refer to the *LDAP* section for more details.

ldap_tls_keyfile

- **type:** Boolean
- **default:** 1
- **Description:** The certificate key file to use when using TLS encryption.

Please refer to the *LDAP* section for more details.

ldap_tls_certfile

- **type:** Boolean
- **default:** 1
- **Description:** The certificate file to use when using TLS encryption.

Please refer to the *LDAP* section for more details.

manage_dhcp

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, Cobbler will rewrite the dhcpd.conf file based on the template `/etc/cobbler/dhcp.template`. If you are using static IP addresses for interfaces, you must enable this option so that static lease entries are written and available for the PXE phase of the installation.

Alternatively, if DNSMASQ is being used for DNS/DHCP, it will manage those configuration files.

Please refer to the *DHCP* section for more details.

manage_dns

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, Cobbler will write the named.conf and BIND zone files based on templates and other settings.

Alternatively, if DNSMASQ is being used for DNS/DHCP, it will manage those configuration files.

Please refer to the *DNS* section for more details.

manage_forward_zones

- **type:** List of Strings
- **default:** []
- **Description:** If enabled along with the manage_dns option, Cobbler will generate configurations for the forward-based zones specified in the list.

Please refer to the *DNS* section for more details.

manage_reverse_zones

- **type:** List of Strings
- **default:** []
- **Description:** If enabled along with the `manage_dns` option, Cobbler will generate configurations for the reverse-based zones specified in the list.

Please refer to the *DNS* section for more details.

manage_rsync

- **type:** Boolean
- **default:** 0
- **Description:** If set, Cobbler will generate the `rsyncd.conf` configuration file. This is required if using a system running cobblerd as a replica master.

Please refer to the *Replication* section for more details.

manage_tftpd

- **type:** Boolean
- **default:** 1
- **Description:** If set, Cobbler will copy files required for the PXE netboot process to the TFTPD root directory and will also generate PXE boot configuration files for systems and profiles.

Please refer to the *Configuration Management* section for more details.

mgmt_classes

- **type:** List of Strings
- **default:** []
- **Description:** A default list of management class names to give all objects, for use with configuration management integration.

Please refer to the *Configuration Management* section for more details.

mgmt_parameters

- **type:** Dictionary
- **default:** {'from_cobbler': 1}
- **Description:** A default list of management parameters to give all objects, for use with configuration management integration.

Please refer to the *Configuration Management* section for more details.

next_server

- **type:** String
- **default:** "127.0.0.1"
- **Description:** If manage_dhcp is enabled, this will be the default next-server value passed to systems that are PXE booting. This value can be overriden on a per-system basis via the `--server` option.

Please refer to the *Multi-Homes cobbler servers* section for more details.

power_management_default_type

- **type:** String
- **default:** "ipmitool"
- **Description:** The default power management type, when using Cobbler's power management feature.

Please refer to the *Power Management* section for more details.

power_template_dir

- **type:** String
- **default:** "/etc/cobbler/power"
- **Description:** The path to the directory containing templates that will be used for generating data sent to the various power management functions (typically provided by cluster fencing agents). As of 2.2.3, templates are no longer required for the default function of most fence agents.

Please refer to the *Power Management* section for more details.

puppet_auto_setup

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, Cobbler will install and configure the Puppet configuration management (https://puppet.com/solutions/configuration-management) software on new systems.

Please refer to the *Puppet Integration* section for more details.

puppetca_path

- **type:** String
- **default:** "/usr/sbin/puppetca"
- **Description:** The path to the puppetca command, which is used by cobbler to auto-register and cleanup Puppet CA certificates during the build process for new systems.

Please refer to the *Puppet Integration* section for more details.

pxe_just_once

- **type:** Boolean
- **default:** 0

- **Description:** If enabled, Cobbler will set the netboot_enabled flag for systems to 0 when the build process is complete. This prevents systems from ending up in a PXE reboot/installation loop which can happen when PXE is set to the default boot option.

Note: This requires the use of the `$SNIPPET('kickstart_done')` in your %post (usually the last line of the `%post` script). This snippet is included in the `sample*.ks` files, so review those as a reference for use.

---

pxe_template_dir

- **type:** String
- **default:** "/etc/cobbler/pxe"
- **Description:** The directory containing the templates used for generating PXE boot configuration files, when `manage_tftpd` is enabled.

redhat_management_key

- **type:** String
- **default:** ""
- **Description:** The default RHN registration key to use with the included RHN/Satellite/Spacewalk registration scripts. This can be overridden on a per-object basis, for instance when you want to use different registration keys to place systems in different RHN channels, etc.

redhat_management_permissive

- **type:** Boolean
- **default:** 0
- **Description:** If set, this will allow per-user access in the Web UI when using the `authn_spacewalk` module for authentication.

However, doing so will permit all Spacewalk/Satellite users with certain roles (`config_admin` and `org_admin`) to edit all of cobbler's configuration. Users should turn this on only if they want this behavior and do not have a cross-multi-org seperation concern. If you have a single org in your satellite, it's probably safe to turn this on to enable the use of the Web UI alongside a Satellite install.

Please refer to the *Spacewalk* section for more details.

redhat_management_server

- **type:** String
- **default:** "xmlrpc.rhn.redhat.com"

- **Description:** The default RHN server to use for registration via the included RHN/Satellite/Spacewalk registration scripts as well as the `authn_spacewalk` authentication module.

Please refer to the *Spacewalk* section for more details.

redhat_management_type

- **type:** String
- **default:** "off"
- **Description:** When using a Red Hat management platform in addition to Cobbler, this option is used to speficy the type of RHN server being used:

```
"off"    : I'm not using Red Hat Network, Satellite, or Spacewalk
"hosted" : I'm using Red Hat Network
"site"   : I'm using Red Hat Satellite Server or Spacewalk
```

Please refer to the *Tips for RHN* section for more details.

register_new_installs

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, this allows `/usr/bin/cobbler-register` (part of the koan package) to be used to remotely add new cobbler system records to cobbler. This effectively allows for registration of new hardware from system records, even during the build process when building a system based only on a profile.

Please refer to the *Auto registration* section for more details.

remove_old_puppet_certs_automatically

- **type:** Boolean
- **default:** 0
- **Description:** If enabled when using Puppet integration, Cobbler can be triggered (through the use of snippets) to automatically remove CA certificates for a given FQDN. This prevents failed Puppet registrations when a conflicting cert already exists.

Please refer to the *Puppet Integration* section for more details.

replicate_rsync_options

- **type:** String
- **default:** "-avzH"

- **Description:** This setting is used to specify additional options that are passed to the rsync command during the replicate process.

Please refer to the *Replication* section for more details.

reposync_flags

- **type:** String
- **default:** "-l -n -d"
- **Description:** This setting is used to specify additional options that are passed to the reposync command during the reposync process. This is specific to yum, and is not used with apt or other repository types.

Please refer to the *Reposync* section for more details.

restart_dhcp

- **type:** Boolean
- **default:** 1
- **Description:** If enabled, Cobbler will restart the dhcpd or dnsmasq daemon during a `cobbler sync` and after all configuration files have been generated. This will only happen when `manage_dhcp` is enabled.

Please refer to the *DHCP* section for more details.

restart_dns

- **type:** Boolean
- **default:** 1
- **Description:** If enabled, Cobbler will restart the named or dnsmasq daemon during a `cobbler sync` and after all configuration files have been generated. This will only happen when `manage_dns` is enabled.

Please refer to the *DNS* section for more details.

restart_xinetd

- **type:** Boolean
- **default:** 1
- **Description:** If enabled, Cobbler will restart the xinetd daemon during a `cobbler sync` and after all configuration files have been generated.

Please refer to the *TFTP* section for more details.

run_install_triggers

- **type:** Boolean
- **default:** 1
- **Description:** If disabled, no install triggers (whether old-style bash or newer python-based scripts) will be run. This is an easy way to lock down cobbler if this functionality is not desired, as these scripts are run as the root user and can present a security risk.

Note: Disabling this will break the `cobbler status` command, which relies on installation triggers to generate the start and stop times for the builds.

Please refer to the *Triggers* section for more details.

scm_track_enabled

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, Cobbler will execute a trigger for all add/edit/sync events which uses the `scm_track_mode` option to revision control Cobbler's data objects.

Please refer to the *Data revision control* section for more details.

scm_track_mode

- **type:** String
- **default:** "git"
- **Description:** If scm_track_enabled is set to true, Cobbler will use the source control method specified by this setting to revision control data objects. Currently, only "git" and "hg" are supported.

Note: Only data in `/var/lib/cobbler` is revision controlled.

Please refer to the *Data revision control* section for more details.

serializer_pretty_json

- **type:** Boolean
- **default:** 0
- **Description:** If enabled, Cobbler will "pretty-print" JSON files that are written to disk, including those for all data object types. By default, the JSON is condensed into a single line, which can make

them a bit difficult to read. The trade-off is a slightly larger file per object (though this size difference is negligable).

server

- **type:** String
- **default:** "127.0.0.1"
- **Description:** This is the address of the cobbler server. As it is used by systems during the install process, it must be the address or hostname of the system as those systems can see the server. If you have a server that appears differently to different subnets (dual homed, etc), you can use the `--server` option to override this value.

This value is also used by the cobbler CLI command, unless the `client_use_localhost` setting is enabled.

Please refer to the *Multi-Homes cobbler servers* section for more details.

sign_puppet_certs_automatically

- **type:** Boolean
- **default:** 0
- **Description:** If enabled when using Puppet integration, Cobbler can be triggered (through the use of snippets) to automatically register CA certificates for a given FQDN, allowing puppet to be run during the `%post` section of the installation without issues.

Please refer to the *Puppet Integration* section for more details.

snippetsdir

- **type:** String
- **default:** "/var/lib/cobbler/snippets"
- **Description:** The default directory containing Cobbler's snippets. Any snippet referenced by the `$SNIPPET('')` call in a template must live under this directory, for security purposes. Snippets can be located in sub-directories here to aid in organization.

template_remote_kickstarts

- **type:** Boolean
- **default:** 0
- **Description:** If this option is enabled and a remote (non-local) kickstart file is specified for an object, Cobbler will fetch the file contents internally and serve a templated version of the file to the client. By default, Cobbler simply passes the remote URL directly to the client.

virt_auto_boot

- **type:** Boolean
- **default:** 1
- **Description:** If enabled, any VM created by Koan will be set to start at boot time.

Please refer to the https://koan.readthedocs.io/ section for more details.

webdir

- **type:** String
- **default:** "/var/www/cobbler"
- **Description:** The directory in which Cobbler will write all of its distribution, repo, and other web-related data.

xmlrpc_port

- **type:** Integer
- **default:** 25151
- **Description:** The port on which cobblerd will listen for XMLRPC connections, in connection with the address/hostname specified in the server setting.

The cobbler CLI command also relies upon this option for connecting to cobblerd unless the `client_use_localhost` setting is enabled.

yum_distro_priority

- **type:** Integer
- **default:** 1
- **Description:** The default yum repo priority for repos managed by Cobbler. If different repos provide the same package name, the one with the lower priority will be used by default. The lower the priorty number, the higher the priority (1 is the highest priority).

This option is only valid for yum repos, and is not used for apt or other repo types.

Please refer to the *Package Management and Mirroring* section for more details.

yum_post_install_mirror

- **type:** Boolean
- **default:** 1

- **Description:** If enabled, Cobbler will add yum.repos.d entries for all repos allocated to a system or profile. If disabled, these repos will only be used during the build process. Normally, this option should be left enabled unless you are using other configuration management systems to configure the repos in use after the build process is complete.

yumdownloader_flags

- **type:** String
- **default:** "–resolve"
- **Description:** Extra flags for the yumdownloader command, which is used to pull down individual RPM files out of a yum repo.

Please refer to the *Package Management and Mirroring* section for more details.

3.4 Managing Services with Cobbler

3.4.1 DHCP

You may want cobbler to manage the DHCP entries of its client systems. It currently supports either ISC DHCP or dnsmasq (which, despite the name, supports DHCP). Cobbler can also be used to manage your DNS configuration (see *DNS* for more details).

To use ISC, your `/etc/cobbler/modules.conf` should contain:

```
[dhcp]
module = manage_isc
```

To use dnsmasq, it should contain:

```
[dns]
module = manage_dnsmasq

[dhcp]
module = manage_dnsmasq
```

Note: Using dnsmasq for DHCP requires that you use it for DNS, even if you disable `manage_dns` in your *Cobbler Settings*. You should not try to mix the ISC module with the dnsmasq module.

You also need to enable such management; this is done in *Cobbler Settings*.

```
manage_dhcp: 1
restart_dhcp: 1
```

The relevant software packages also need to be present; *Check* will verify this.

When To Enable DHCP Management

DHCP is closely related to PXE-based installation. If you are maintaining a database of your systems and what they run, it can make sense also to manage hostnames and IP addresses. Controlling DHCP from Cobbler can coordinate all this. This capability is a good fit if you can control DHCP for a lab or datacenter and want to run DHCP from the same server where you are running Cobbler. If you have an existing configuration of things that cobbler shouldn't be managing, you can copy them into your `/etc/cobbler/dhcp.template`.

The default behaviour is for cobbler **not** to manage your DHCP infrastructure. Make sure that in your existing `dhcp.conf` the next-server entry and filename information are correct to serve up `pxelinux.0` to the machines that want it (for the case of bare metal installations over PXE).

Setting up

ISC considerations

The master DHCP file when run from cobbler is `/etc/cobbler/dhcp.template`, not the more usual `/etc/dhcpd.conf`. Edit this template file to suit your environment; this is mainly just making sure that the DHCP information is correct. Youcan also include anything you may have had from an existing setup.

DNSMASQ considerations

If using dnsmasq, the template file is `/etc/cobbler/dnsmasq.template` but it basically works as for ISC (above). Remember that dnsmasq also provides DNS.

How It Works

Suppose the following command is given (where `<profile name>` is an existing profile in cobbler):

```
$ cobbler system add --name=foo --profile=<profile name>
  --interface=eth0 --mac=AA:BB:CC:DD:EE:FF --ip-address=192.168.1.1
```

That will take the template file in `/etc/cobbler/dhcp.template`, fill in the appropriate fields, and generate a fuller configuration file `/etc/dhcpd.conf` that includes this machine, and ensures that when AA:BB:CC:DD:EE:FF asks for an IP, it gets 192.168.1.1. The `--ip-address=...` specification is optional; DHCP can make dynamic assignments within a configured range.

To make this active, run:

```
$ cobbler sync
```

As noted in the *Sync* section, managing DHCP with the ISC module is one of the few times you will need to use the full sync via `cobbler sync`.

Itanium: additional requirements

Itanium-based systems are more complicated and special the other architectures, because their bootloader is not as intelligent, and requires a "filename" value that references elilo, not pxelinux.

- When creating the distro object, make sure that `--arch=ia64` is specified.
- You need to create system objects, and the `--mac-address` argument is mandatory. (This is due to a deficiency in LILO where it will ask for an encoded IP address, but will not ask for a PXE configuration file based on the MAC address.)
- You need to specify the `--ip-address=...` value on system objects.
- In `/etc/cobbler/settings`, you must (for now) choose `dhcp_isc`.

Also, sometimes Itaniums tend to hang during net installs; the reasons are unknown.

ISC and OMAPI for dynamic DHCP updates

OMAPI support for updating ISC DHCPD is actually not supported. This was a buggy feature (we think OMAPI itself is buggy) and apparently OMAPI is slated for removal in a future version of ISC dhcpd.

Static IPs

Lots of users will deploy with DHCP for PXE purposes and then use the Anaconda installer or other mechanism to configure static networking. For this, you do not need to use this DHCP Management feature. Instead you can configure your DHCP to provide a dynamic range, and configure the static addresses by other mechanisms.

For instance `cobbler system ...` can set each interface. Cobbler's default *Snippets* will handle the rest.

Alternatively, if your site uses a *Configuration Management* system, that might be suitable for such configuration.

If You Don't Have Any DHCP

If you don't have any DHCP at all, you can't PXE, and you can ignore this feature, but you can still take advantage of *Build ISO* for bare metal installations. This is also good for installing machines on different networks that might not have a next-server configured.

3.4.2 DNS

You may want cobbler to manage the DNS entries of its client systems. Cobbler can do so automatically by using templates. It currently supports either dnsmasq (which also provides DHCP) or BIND. Cobbler also has the ability to handle *DHCP*.

To use BIND, your `/etc/cobbler/modules.conf` should contain:

```
[dns]
module = manage_bind

[dhcp]
module = manage_isc
```

To use dnsmasq, it should contain:

```
[dns]
module = manage_dnsmasq

[dhcp]
module = manage_dnsmasq
```

You should not try to mix these.

You also need to enable such management; this is done in `/etc/cobbler/settings`:

```
manage_dns: 1

restart_dns: 1
```

The relevant software packages also need to be present; `cobbler check` will verify this.

General considerations

- Your maintenance is performed on template files. These do not take effect until a `cobbler sync` has been performed to generate the run-time data files.
- The serial number on the generated zone files is the cobbler server's UNIX epoch time, that is, seconds since 1970-01-01 00:00:00 UTC. If, very unusually, your server's time gets reset backwards, your new zone serial number could have a smaller number than previously, and the zones will not propagate.

BIND considerations

In `/etc/cobbler/settings` you will need entries resembling the following:

```
manage_forward_zones: ['foo.example.com', 'bar.foo.example.com']

manage_reverse_zones: ['10.0.0', '192.168', '172.16.123']
```

Note that the reverse zones are in simple IP ordering, not in BIND-style "0.0.10.in-addr.arpa".

(??? CIDR for non-octet netmask ???)

Restricting Zone Scope

DNS hostnames will be put into their "best fit" zone. Continuing the above illustration, example hosts would be placed as follows:

- baz.bar.foo.example.com as host baz in zone bar.foo.example.com
- fie.foo.example.com as host fie in foo.example.com
- badsub.oops.foo.example.com as host badsub.oops in foo.example.com

Default and zone-specific templating

Cobbler will use /etc/cobbler/bind.template and /etc/cobbler/zone.template as a starting point for BIND's named.conf and individual zone files, respectively. You may drop zone-specific template files into the directory /etc/cobbler/zone_templates/ which will override the default. For example, if you have a zone foo.example.com, you may create /etc/cobbler/zone_templates/foo.example.com which will be used in lieu of the default /etc/cobbbler/zone.template when generating that zone. This can be useful to define zone-specific records such as MX, CNAME, SRV, and TXT.

All template files must be user edited for the local networking environment. Read the file and understand how BIND works before proceeding.

BIND's *named.conf* file and all zone files will be updated only when "cobbler sync" is run, so it is important to remember to use it.

Other

Note that your client's system interfaces _must_ have a --dns-name set to be considered for inclusion in the zone files. If cobbler system report shows that your --dns-name is unset, it can be set by:

```
cobbler system edit --name=foo.example.com --dns-name=foo.example.com
```

You can set a different such name per interface and each will get its own respective DNS entry.

DNSMASQ considerations

You should review and adjust the contents of /etc/cobbler/dnsmasq.template.

3.4.3 rsync

3.4.4 TFTP

3.5 Kickstart Templating

Kickstarts automate Red Hat and Fedora based Linux installations. This document also pertains to templating as it relates to other distributions, which also have similar answer files.

With any distribution if you are repeatedly installing a lot of different configurations, you can get into a scenario where you have too many different answer files to maintain or the answer files get to be rather

complex and unruly. A kickstart templating system, such as Cobbler's, can help you keep these under control.

When cobbler profiles (created with `cobbler profile add`) are created with the parameter `--kickstart` and the kickstart lives on the filesystem, the file is treated as a kickstart template, not a raw kickstart. This means they contain information about how to build a kickstart, but a given template might be modified by various variables on a per profile or per system basis. This allows you to use the same source information (and maintain that), but have it work in different ways for different profiles and different systems.

It also means that Cobbler, when you use `cobbler import` can assign kickstarts that work for fully automated installs out of the box, but they are also easy to customize and change by manipulating settings on cobbler objects or editing the global cobbler settings file.

The main feature in cobbler kickstart templating is the ability to declare variables on cobbler objects like distributions, profiles, and systems. On the command line, this is the `--ksmeta` parameter. The same source templates can result in different output based on different variables set in cobbler with `--ksmeta`, which this document will explain.

3.5.1 Down The Rabbit Hole

Templating is very powerful, especially if you have a very large amount of profiles and/or systems to maintain.

Users can just treat templates like kickstarts, not explore this feature fully, or pretend this feature doesn't exist, but if more customization is needed, the support is there.

This article will get into a lot of complicated (and optional) topics, so just take from it what you need. If this sounds too complicated, it's not required that you understand all of it.

3.5.2 First off

Cobbler uses Cheetah (https://cheetahtemplate.org/) for its kickstart templating engine. Read more at the Cheetah page and you will learn a lot of advanced features you can employ in your kickstart templates. However, knowledge of all parts of Cheetah are not required. Basic variable solution is automagic and doesn't require any advanced knowledge of how the templating works.

Cheetah is nice in that most of the things you can do in the powerful Python programming language you can do in your Cheetah templates, so it is very flexible. However some things do not look quite like Python, so you have to pay a little attention to the syntax.

3.5.3 Hierarchy

As with all things in cobbler, kickstart template variables (`--ksmeta`) take the following precendence:

- Distro parameters come first
- Profile parameters override Distro Parameters
- System parameters override Profile Parameters

If a distro has an attribute "bar" and the profile that is the child of the distro does not, the profile will still have the value of "bar" accessible in the kickstart template. However, if the system then supplies it's own value for "bar", the system value, not the distro value, will be used in rendering the kickstart for the system. This is what we mean by "inheritance". It is a simple system of overriding things from the most specific (systems) to the least specific (profiles, then distros) to allow for customizations.

3.5.4 Basic Variable substitution

Given the following commands

```
cobbler profile add --name=foo --distro=RHEL-6-x86_64 --kickstart=/var/lib/
↪cobbler/kickstarts/sample.ks
cobbler profile edit --name=foo --ksmeta="noun=spot verb=run"
cobbler system add --name=bar --profile=foo --ksmeta="verb=jump"
```

And the kickstart template `/var/lib/cobbler/kickstarts/sample.ks`:

```
See $noun $verb
```

The following file would be generated for profile foo:

```
See spot run
```

And for the system, bar:

```
See spot jump
```

This is of course very contrived, though you can imagine substituting in things like server locations, configuration file settings, timezones, etc.

3.5.5 Snippets

If you find yourself reusing a lot of pieces of code between several different kickstart templates, cobbler snippets are for you.

Read more at *Snippets*.

That page also includes some user contributed snippet examples – some of which make some heavy use of the Cheetah template engine. Snippets don't have to be complex, but you may find those examples educational.

3.5.6 Escaping

If your kickstart file contains any shell macros like `$(list-harddrives)` they should be escaped like this:

```
$(list-harddrives)
```

This prevents Cheetah from trying to substitute something for `$(list-harddrives)`, or worse, getting confused and erroring out.

In general, escaping things doesn't hurt, even though in all cases, things don't have to be escaped.

Example:

```
This is an $elephant
```

If there was no kickstart variable for "elephant", the kickstart templating engine would leave the string as is ... `$elephant`

You should also be careful of the following stanza:

```
#start some section (this is a comment)
echo "doing stuff"
#end some section (this is a comment)
```

if you want a comment to start with the word "end" place a space after the "#" like this:

```
# start some section (this is a comment)
echo "doing stuff"
# end some section (this is a comment)
```

3.5.7 Built In Variables

Cobbler includes a lot of built in kickstart variables.

What variables can I use?

Run this command to see all the templating variables at your disposal.

```
cobbler system dumpvars --name=system
```

Some of the built in variables that can be useful include `$mac_address`, `$ip_address`, `$distro`, `$profile`, `$hostname`, and so forth. You will recognize these as being commands that you would see in cobbler command line options.

To make this a bit more clear, look at the following system add command:

```
cobbler system add --name=spartacus --profile=f10webserver-i386 --ip-
↪address=192.168.50.5 --mac=AA:BB:CC:DD:EE:FF --hostname=spartacus.example.
↪org
```

For the above command, assuming the kickstart template for fc6webserver contained the following line:

```
echo "I was installed from Cobbler server $server and my system name is
↪$system_name" > /etc/motd
```

The above line would be rendered as:

```
"I was installed from Cobbler server cobbler.example.org and my system name␣
↪is spartacus"
```

Again, the examples above are a bit contrived, but you can see how every variable given to the command line is accessible within templating. This is a rather useful feature and prevents having to specify a lot of additional templating variables with `--ksmeta`.

3.5.8 Checking For Variables That Might Not Exist

Suppose you have some system objects that define a value for "foo", but sometimes they don't.

The following Cheetah templating trick can be used to access a variable if it exists, and assign a default value if it doesn't exist.

```
#set $selinux_mode = $getVar('selinux', 'enforcing')
```

or just

```
$getVar('selinux', 'enforcing')
```

As a corollary, if you need to include a specific line in a kickstart file only if a variable is defined, that is also doable.

```
#if $foo
  this line will show $foo but only if it is defined, else there will be␣
↪nothing here
#end if
```

3.5.9 Networking

Cobbler actually handles templating around network setup for you, via some rather clever snippets used in files such as `/var/lib/cobbler/kickstarts/sample.ks`

However, if you need to access networking information from systems in Cobbler templating, you do it as follows:

```
$system.interfaces['eth1']['mac_address']
```

This should also be apparent in the output from `cobbler system dumpvars --name=foo`

Again, usually you should not have to access these directly, see *Advanced Networking* for details about Cobbler templates all the network info out for you.

3.5.10 Built-in functions and extensibility

You can optionally expose custom-written functions to all Cheetah templates. To see a list of these functions you have configured for your site (Cobbler doesn't currently ship with any) and/or add new functions, see *Extending cobbler*.

3.5.11 Raw Escaping

Cobbler uses Cheetah (https://cheetahtemplate.org) for kickstart templating. Since Cheetah sees "$" as "include this variable", it is usually a good idea to escape dollar signs in kickstart templates with \$. However, this gets to be hard to read over time. It is easier to declare a block "raw", which means it will not be evaluated by Cheetah.

```
#raw
This $dollar sign will stay in the output regardless of what the --ksmeta␣
↪metadata variables are
#end raw
```

It is possible to cheat by assigning bash variables from the values of Cheetah variables, and use them inside raw blocks. This is useful if you want your shell scripts to be able to access templating variables but don't really want to make sure escaping is all super-correct.

```
%pre
foo = $foo
#raw
This $foo will be evaluated and will not appear with a dollar sign
and if you included funky shell scripts here you wouldn't have to worry
about escaping anything.  The $foo comes from bash and not Cheetah
#end raw
```

Raw escaping and Snippets

Be aware: raw escaping also applies to SNIPPET directives. For example:

```
#raw
$SNIPPET('my_snippet')
#end raw
```

Will not work as expected. The result will be:

```
$SNIPPET('my_snippet')
```

Because `$SNIPPET` is inside `#raw #end raw`, Cheetah ignores it, and the snippet is not included. Note this also applies to the legacy `SNIPPET::` syntax.

The `#raw #end raw` directives should instead be placed inside of `my_snippet`.

3.5.12 Conditionals

Cheetah supports looping and if statements. For more of this, see the Cheetah (http://cheetahtemplate.org) web page.

(This section needs to be expanded)

3.5.13 "Stanza" Support

Stanzas are the precursor to Cobbler snippets. Certain built-in complex pieces of code are auto-generated by Cobbler, from within the Cobbler source code, that vary based upon the configuration of the cobbler object being rendered. These sections are not user extensible, unlike the newer snippet support. These are being explained here to give folks an idea of why they should leave these weird dollar sign variables in their kickstarts, but in general, more cobbler stanzas will not be added. The new snippets are the user-extensible way to go.

Certain blocks of kickstart code are substituted for the following variables:

- `$yum_repo_stanza` – this is replaced with the code neccessary to set up any repos that are associated with the given cobbler profile, for use during install time. This should be present towards the top of a kickstart, but only for kickstarts that are RHEL5 and later or FC6 and later. Before those versions, kickstart/Anaconda did not support the "repo" directive.
- `$yum_config_stanza` – this is replaced with the code neccessary to configure the installed system to use the yum repos set up during install time for regular operation. In other words, it sets up `/etc/yum.repos.d` on the provisioned system. This works for all machines that can have yum installed. If the value in `/var/lib/cobbler/settings` for `yum_post_install_mirror` is set, in addition, the provisioned system will be pointed to the boot server as an install source for "core" packages as well as any additional repos.
- `$kickstart_done` – this is replaced with a specially formatted `wget`, that places an entry in the cobbler and/or Apache (depending on how implemented at the time) log file, allowing `cobbler status` to better tell when kickstarts are fully complete. The implementation of what `kickstart_done` means may vary depending on the cobbler version, but it should always be placed in a kickstart template as the last line in %post. Beware in version 2.2.0, `$kickstart_done` does not exist anymore. Use `$SNIPPLET('kickstart_done')` instead between a cheetah stanza.
- (there may be other *Snippets* and macros used not listed above)

Over time these will become first class Cobbler snippets.

3.5.14 Validation

Cobbler contains a command `cobbler validateks` that will run ksvalidator (part of the pykickstart package) against all rendered kickstarts to see if Anaconda will likely like them. It should be noted that ksvalidator is not perfect, and in some cases, it will report false positives and/or negatives. However, it is still useful to make sure that your rendered output from the kickstart templates is still good.

Testing an install in a VM is often a better idea.

3.5.15 Looking at results

As was said earlier, what is provided for `--kickstart` is a template, not a kickstart. Templates are used to generate kickstarts. The actual contents of the files are served up dynamically from Python and Apache. If you would like to see the output of cobbler first hand (for your own review), you can run the following commands:

For profiles: `cobbler profile getks --name=profile-name`

For systems: `cobbler system getks --name=system-name`

3.5.16 Calling Python Code

Cheetah lets you use python modules from inside the templates.

Example:

```
#import time
$time.strftime('%m/%d/%Y')
```

However what modules you can import are very limited for security reasons. If you see a module cobbler won't let you import, add it to the whitelist in `/etc/cobbler/settings`.

3.5.17 Comments

Cheetah makes comments with double hash marks `##`. Any line starting with `##` will not show up in the rendered kickstart file at all.

Kickstart comments `#` will show up in the rendered output.

Both styles of comments may be mixed. You can use `##` to describe what you are doing in your templates, and those `##` comments won't show up when someone looks at the rendered kickstart file in `/var/www/cobbler`.

If this sounds complicated, it is. It's even more complicated in that Cheetah has special meanings for some things starting with `#` like `#if` or `#include`. It's pretty much safe to just use the single `#` comment form everywhere though.

3.5.18 Further info

This was originally a separate section in "Advanced topics" (itself originally part of the original, oversized man page. It has been moved here, but needs to be merged properly with the text above.

If and only if `--kickstart` options reference filesystem URLs, `--ksmeta` allows for templating of the kickstart files to achieve advanced functions. If the `--ksmeta` option for a profile read `--ksmeta="foo=7 bar=llama"`, anywhere in the kickstart file where the string `$bar` appeared would be replaced with the string `llama`.

To apply these changes, `cobbler sync` must be run to generate custom kickstarts for each profile/system.

For NFS and HTTP kickstart URLs, the `--ksmeta` options will have no effect. This is a good reason to let cobbler manage your kickstart files, though the URL functionality is provided for integration with legacy infrastructure, possibly including web apps that already generate kickstarts.

3.5.19 Other Resources

- Kickstart-list (http://www.redhat.com/mailman/listinfo/kickstart-list) is a great mailing list for info
- Cheetah web page (http://cheetahtemplate.org)
- Article on some Cheetah features (devshed.com) (http://www.devshed.com/c/a/Python/Templating-with-Cheetah/3/)
- RHEL 5 Install Guide (section on kickstart options) (http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/s1-kickstart2-options.html)

3.6 Snippets

Snippets are a way of reusing common blocks of code between kickstarts (though this also works on files other than kickstart templates, but that's a sidenote). For instance, the default Cobbler installation has a snippet called "`$SNIPPET('func_register_if_enabled')`" that can help set up the application called Func.

This means that every time that this SNIPPET text appears in a kickstart file it is replaced by the contents of `/var/lib/cobbler/snippets/func_register_if_enabled`. This allows this block of text to be reused in every kickstart template – you may think of snippets, if you like, as templates for templates!

To review, the syntax looks like: `SNIPPET::snippet_name_here`

Where the name following snippet corresponds to a file in `/var/lib/cobbler/snippets` with the same name.

Snippets are implemented using a Cheetah function. Although the legacy syntax ("`SNIPPET::snippet_name_here`") will still work, the preferred syntax is: `$SNIPPET('snippet_name_here')`

You can still use the legacy syntax if you prefer, but a small bit of functionality is lost compared to the new style.

3.6.1 Advanced Snippets

If you want, you can use a snippet name across many templates, but have the snippet be different for specific profiles and/or system objects. Basically this means you can override the snippet with different contents in certain cases.

An example of this is if you want to a snippet called "`partition_select`" slightly for a certain profile (or set of profiles) but don't want to change the master template that they all share.

This could also be used to set up a package list – for instance, you could store the base package list to install in `/var/lib/cobbler/snippets/package_list`, but override it for certain profiles and systems. This would allow you to ultimately create less kickstart templates and leverage the kickstart templating engine more by just editing the smaller and more easily readable snippet files. These also help keep your kickstarts manageable and keep them from becoming too long.

The resolution order for kickstart templating evaluation is to use the following paths in order, finding the first one if it exists (`per_distro` was added in cobbler 2.0).

- `/var/lib/cobbler/snippets/per_system/$snippet_name/$system_name`
- `/var/lib/cobbler/snippets/per_profile/$snippet_name/$profile_name`
- `/var/lib/cobbler/snippets/per_distro/$snippet_name/$distro_name`
- `/var/lib/cobbler/snippets/$snippet_name`

As with the rest of cobbler, systems override profiles as they are more specific, though if the system file did not exist, it would use the profile file. As a general safeguard, always create the `/var/lib/cobbler/snippets/$snippet_name` file if you create the `per_system` and `per_profile` ones.

3.6.2 Subdirectories

Snippets can be placed in subdirectories for better organization, and will follow the order of precedence listed above. For example: `/var/lib/cobbler/snippets/$subdirectory/$snippet_name`

This would be referenced in your kickstart template as `$SNIPPET('$subdirectory/$snippet_name')`.

Replace the dollar sign names with the actual values, such as ... `$SNIPPET('example.org/dostuff')`

3.6.3 Variable Snippet Names

Sometimes it is useful to determine which snippet to include at render time. In Cobbler 1.2, this can be done by omitting the quotes and using a template variable: `$SNIPPET($my_snippet)`

Note that this DOES NOT work with the legacy syntax:

```
#set my_snippet = 'foo'
SNIPPET::$my_snippet
```

This will not behave as expected. We would like it to include a snippet named 'foo'; instead, it will search for a snippet named `$my_snippet`.

3.6.4 Cobbler SNIPPETs versus Cheetah #include

It seems that `$SNIPPET` and `#include` accomplish the same thing. In fact, `$SNIPPET` uses `#include` in its implementation! While the two perform similar tasks, there are some important differences:

- `$SNIPPET` will search for profile and system-specific SNIPPETS (See Advanced Snippets)
- `$SNIPPET` will include the namespace of the snippet, so any functions `#def`'ed in the snippet will be accessible to the main kickstart file. `#include` will not do this.

For example:

```
my_snippet:

    #def myfunc()
    ...
    #end def

my_kickstart:

    #include '/var/lib/cobbler/snippets/my_snippet'  ## UGH!
    $myfunc()
```

Will NOT work. However,

```
my_kickstart:

    $SNIPPET('my_snippet') ## Much better!
    $myfunc()
```

Will work as expected. It will search for the snippet itself, and it will make sure `myfunc()` is callable from `my_kickstart`.

3.6.5 Scoping issues

Cobbler uses Cheetah to implement snippets, so variables in these snippets are subject to Cheetah's scoping rules (except `#def`'ed functions). Variables set inside a snippet cannot be accessed in the main kickstart file. For example:

```
my_snippet:

    #set dns1 = '192.168.0.1'

my_kickstart:

    $SNIPPET('my_snippet')
    echo 'nameserver $dns1' >> /etc/resolv.conf
```

Will not work as expected. The variable `$dns1` is destroyed as soon as Cheetah finishes processing `my_snippet`. To fix this, use the 'global' modifier:

```
my_snippet:

    #set global dns1 = '192.168.0.1'
```

Note that the 'global' modifier is not needed on `#def` directives. In fact, '`#def global`' is a syntax error in Cheetah.

3.6.6 Recursive or Nested Snippets

Cobbler Snippets can allow for nested snippets. For example:

```
my_kickstart:

    Main content
    $SNIPPET('my_snippet')
    More main content

my_snippet:

    Snippet content
    $SNIPPET('my_subsnippet')
    More snippet content

my_subsnippet:

    Subsnippet content
```

Will yield:

```
Main content
Snippet content
Subsnippet content
More snippet content
More main content
```

as expected.

3.6.7 Kickstart Snippet Cookbook

The rest of this page contains Snippets contributed by users of Cobbler that provide examples of usage and some quick recipes that can be used/extended for your environment.

If you come up with any clever tricks, paste them here to share, and also share them with the cobbler mailing list so we can talk about them.

Note that some of these rely on cobbler's Cheetahtemplate (http://cheetahtemplate.org) *Kickstart Templating* engine pretty heavily so they might be a little hard to read at first. Snippets can just be simple reusable blocks of basic copy and paste text and can also be simple. Either way works depending on what you want to do.

Note: Content provided here is not part of Cobbler's "core" code so we may not be able to help you on the mailing list or IRC with snippets that aren't yet part of cobbler's core distribution. Cobbler does ship a few in `/var/lib/cobbler/snippets` that we can answer questions on, and in general, if you have a good idea, we'd love to work with you to get it shipped with Cobbler.

Adding an SSH key to authorized keys

```
# Install Robin's public key for root user
cd /root
```

(continues on next page)

(continued from previous page)

```
mkdir --mode=700 .ssh
cat >> .ssh/authorized_keys << "PUBLIC_KEY"
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtDHt4p16wtfUeyzyWBN7R1SXcnjq+R/
↪ojQmiv8HOfYPNM48eCXYdCiNHD4tPCxuizLulqq1zG06B2OPVy9GXXtyXcAXLAQdGaZwDdKU6gHMUplUChSyDpXK
↪robin@robinbowes.com
PUBLIC_KEY
chmod 600 .ssh/authorized_keys
```

Instructions for setup:

1. Decide what to call your snippet. I'll use the name *publickey_root_robin*.
2. Save your code in */var/lib/cobbler/snippets/<snippet name>*
3. Add your new snippet to your kickstart template, e.g.

```
%post
SNIPPET::publickey_root_robin
$kickstart_done
```

Disk Configuration

Contributed by: Matt Hyclak

This snippet makes use of if/else, getVar, and the `split()` function.

It provides some additional options for partitioning compared with the example shipped with Cobbler. If the disk you want to partition is not sda, then simply set a ksmeta variable for the system (e.g. `cobbler system edit --name=oldIDEbox --ksmeta="disk=hda"`)

```
#set $hostname = $getVar('$hostname', None)
#set $hostpart = $getVar('$hostpart', None)
#set $disk = $getVar('$disk', 'sda')

#if $hostname == None
#set $vgname = "VolGroup00"
#else
#if $hostpart == None
#set $hostpart = $hostname.split('.')[0]
#set $vgname = $hostpart + '_sys'
#end if
#end if

clearpart --drives=$disk --initlabel
part /boot --fstype ext3 --size=200 --asprimary
part swap --size=2000 --asprimary
part pv.00 --size=100 --grow --asprimary
volgroup $vgname pv.00
logvol / --vgname=$vgname --size=16000 --name=sysroot --fstype ext3
logvol /tmp --vgname=$vgname --size=4000 --name=tmp --fstype ext3
logvol /var --vgname=$vgname --size=8000 --name=var --fstype ext3
```

(continues on next page)

(continued from previous page)

```
#if $hostpart == "bing"
logvol /var/www --vgname=$vgname --size=16000 --name=www
#else if $hostpart == "build32"
logvol /var/fakedirectory --vgname=$vgname --size=123456789 --name=fake
#end if
```

Another partitioning example

Use software raid if there are more then one disk present (e.g. `cobbler system edit --name=webServer --ksmeta="disks=sda,sdb"`)

Contributed by: Harry Hoffman

```
#set disks = $getVar('$disks', 'sda')
#set count = len($disks.split(','))

#if $count >= 2
part /boot --fstype ext3 --size=100 --asprimary --ondisk=${disks.split(',
↪')[0]}
part /boot2 --fstype ext3 --size=100 --asprimary --ondisk=${disks.split(',
↪')[1]}
part swap --size=1024 --asprimary --ondisk=${disks.split(',')[0]}
part swap --size=1024 --asprimary --ondisk=${disks.split(',')[1]}

part raid.10 --size=1 --grow --ondisk=${disks.split(',')[0]}
part raid.11 --size=1 --grow --ondisk=${disks.split(',')[1]}
raid pv.01 --fstype "physical volume (LVM)" --level=RAID1 --device=md0 raid.
↪10 raid.11
#else
part /boot --fstype ext3 --size=100 --asprimary --ondisk=${disks.split(',
↪')[0]}
part swap --size=1024 --asprimary --ondisk=${disks.split(',')[0]}
part pv.01 --size=1 --grow --ondisk=${disks.split(',')[0]}
#end if

volgroup internal_hd --pesize=32768 pv.01

logvol / --name=slash --vgname=internal_hd --fstype ext3 --size=4096
logvol /tmp --name=tmp --vgname=internal_hd --fstype ext3 --size=1024
logvol /var --name=var --vgname=internal_hd --fstype ext3 --size=8192
logvol /usr --name=usr --vgname=internal_hd --fstype ext3 --size=8192
```

Package Selection by hostname

Contributed by: Matt Hyclak

Note: Advanced Snippets in all recent versions of Cobbler make this unneccessary (this is an older snippet),

but it's still a neat trick to learn some Cheetah skills.

This snippet makes use of if/else, getVar, the `split()` function, include, and try/except.

This snippet allows the administrator to create a file containing the package selection based on hostname and includes it if possible, otherwise it fallse back to a default.

```
#set $hostname = $getVar('$hostname', None)

#if $hostname == None
%packages
@base
#else
#set $hostpart = $getVar('$hostpart', None)
#if $hostpart == None
#set $hostpart = $hostname.split('.')[0]
#end if
#set $sourcefile = "/var/lib/cobbler/packages/" + $hostpart

%packages
#try
  #include $sourcefile
#except
@base
#end try
#end if
```

Package Selection by profile name

Contributed by: Luc de Louw

This snippet add or removes packages depending on the profile name. Assuming you have profiles named rhel5, rhel5-test, rhel4 and rhel4-test. You need to install packages depending if it a test system or not.

```
#if 'test' in $profile_name
#Test System selected, adding some more packages
compat-gcc-32
compat-gcc-32-c++
compat-libstdc++-296
compat-libstdc++-33.i386
compat-libstdc++-33.x86_64
libstdc++.i386
libstdc++.x86_64

#else
#Non-test System detected, removing some packages
-openmotif

#end if
```

Add `$SNIPPET('snippetname')` at the `%packages` section in the kickstart template

Root Password Generation

Contributed by: Matt Hyclak

This snippet makes use of if/else, getVar, and demonstrates how to import and use python modules directly.

This snippet generates a password from a pattern of the first 4 characters of the hostname + "andsomecommonpart", creates an appropriate encrypted string with a random salt, and outputs the appropriate rootpw line. (mdehaan warns – this snippet isn't secure as the variable 'hostname' can still be easily read from Cobbler XMLRPC, if systems have access to it. Credentials are NOT required to read metadata variables like the hostname, and in this case, the hostname isn't hard to guess either)

```
#set $hostname = $getVar('$hostname', None)

#if $hostname
#set $distinct = $hostname[0:4]
#set $rootpw = $distinct + "andsomecommonpart"

#from crypt import crypt
#from whrandom import choice
#import string

#set $salt_pop = string.letters + string.digits + '.' + '/'
#set $salt = ''

#for $i in range(8)
#set $salt = $salt + choice($salt_pop)
#end for

#set $salt = '$1$' + $salt
#set $encpass = crypt($rootpw, $salt)
rootpw --iscrypted $encpass
#end if
```

VMWare Detection

Contributed by: Matt Hyclak

This snippet makes use of if/else, getVar, and demonstrates how to make multiple comparisons in an if statement.

This snippet detects if the host is a VMWare guest, and adds a special kernel repository.

```
#set $mac_address = $getVar('$mac_address', None)
#if $mac_address
#set $mac_prefix = $mac_address[0:8]

#if $mac_prefix == "00:0c:29" or $mac_prefix == "00:05:69" or $mac_prefix ==
↪"00:50:56"

cat << EOF >> /etc/yum.repos.d/vmware-kernels.repo
[vmware-kernels]
```

(continues on next page)

(continued from previous page)

```
name=VMWare 100Hz Kernels
baseurl=http://people.centos.org/~hughesjr/vmware-kernels/4/`uname -m`/
enabled=1
gpgcheck=0
priority=2
EOF

yum -y install kernel

#end if
#end if
```

RHEL Installation Keys

Contributed by: Wil Cooley

RHEL uses keys (also called *Installation Number*) to determine the appropriate packages to install. To fully automate a RHEL installation, the kickstart needs a *key* option, either setting the key or explicitly skipping it.

This is not to be confused with *Tips for RHN*, which includes registration instructions for RHN Hosted and Satellite. Cobbler actually is happy with "`key --skip`" in most cases.

See also:

- [http://kbase.redhat.com/faq/FAQ_103_8967.shtm{]}(http://kbase.redhat.com/faq/FAQ_103_8967.shtm)
- http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/s1-kickstart2-options.html#id3080516

Add this to the kickstart template:

```
# RHEL Install Key
key $getVar('rhel_key', '--skip')
```

Then you can specify the key in the *ksmeta* system definition:

```
# cobbler system edit --name=00:02:55:fa:6b:2b --ksmeta="rhel_key=xxx"
```

If *rhel_key* is not specified, then it will fall back to *–skip*.

Configure Timezone Based on Hostname

Contributed by: Jeff Schroeder (http://www.digitalprognosis.com)

This snipped will print the correct timezone line for your kickstart based on the system's hostname. It is highly dependent on a consistent naming scheme and will have to be edited for each environment. Using

multiple lines to set the associative array seemed like the sanest way to do this to make adding and removing new locations easy.

```
#if $getVar("system_name","") != ""
    #set foo = {}
    #set foo['nyc'] = 'America/New_York'
    #set foo['lax'] = 'America/Los_Angeles'
    #set foo['sin'] = 'Asia/Singapore'
    #set foo['tyo'] = 'Asia/Tokyo'
    #set foo['syd'] = 'Australia/Sydney'
    #set foo['dc'] = 'America/New_York'
    #set hostname = $getVar('system_name').split('.')
    #import re
## Work on hosts with funky hostnames like test001.lab01.lax07.int
    #if re.match('^lab', $hostname[1])
        #del $hostname[1]
    #end if

    #if $len($hostname) == 3  ## ie: ns1.lax07.int
        #set cluster = re.match('^[a-z]+', $hostname[1].lower()).group()

timezone $foo[cluster]
    #else:
# Could not autodetect hostname
timezone --utc
    #end if
#end if
```

Install HP Proliant Support Pack (PSP)

Contributed by: Dave Hatton

This snippet automatically installs the HP Proliant Support Pack (PSP). You may wish to adjust the location that the tarball is downloaded and uncompressed to, and remove the install package after installation.

```
 mkdir -p /software

 /usr/bin/wget -O /software/psp-8.00.rhel5.x86_64.en.tar.gz http://@@server@@/
↪cblr/localmirror/psp-8.00.rhel5.x86_64.en.tar.gz

 cd /software && /bin/tar -xzf psp-8.00.rhel5.x86_64.en.tar.gz


 /bin/cat >>/etc/rc3.d/S99install-hppsp <<EOF
#!/bin/sh
#This script will install the HP PSP
#set -xv

 /bin/echo "Starting PSP Install: "
 cd /software/compaq/csp/linux && ./install800.sh --nui --silent
```

(continues on next page)

(continued from previous page)

```
 /bin/rm -f $0

 exit 0
EOF

 /bin/chmod 755 /etc/rc3.d/S99install-hppsp
```

3.7 Package Management and Mirroring

3.7.1 About

This is a walkthrough of how to set up cobbler to be a full fledged mirror of everything install and update related that you might ever be interested in.

Updates and package installation are closely related. If you're doing one, it makes sense to do the other.

3.7.2 Why would you be interested in this?

Suppose you manage a large number of machines and are (A) not allowed to get to the outside world, (B) bandwidth constrained, or (C) wanting to get access to 3rd party packages including custom yum repositories.

All of these are good reasons to want a mirror server for all things kickstart and yum related. Cobbler can do that for you.

3.7.3 How-To

The following instructions walk through an example of setting up a mirror of a Fedora install tree, including any updates. This will require a good bit of hard disk space (we'll show you how to hardlink to save space later), so be prepared :). These same commands work for all varieties of RHEL, Fedora, or Centos.

First, follow the setup for a DVD import here using the Fedora 12 install media. See [Using Cobbler Import](Using cobbler import).

Once the import is complete, we'll add the mirrors...

```
$ cobbler repo add --mirror=http://download.fedora.redhat.com/pub/fedora/
↪linux/updates/12/ --name=f12-i386-updates
$ cobbler repo add --mirror=http://download.fedora.redhat.com/pub/fedora/
↪linux/releases/12/Everything/i386/ --name=f12-i386-everything
```

Please replace i386 with your preferred architecture. If you own x86_64 or ppc machines as well, just change it. If you're not running Fedora, insert your yum URLs of choice. It all works the same!

These are just a few common examples. Say you have a RHEL5 or Centos4 machine? Perhaps you would want to add something from EPEL (http://fedoraproject.org/wiki/EPEL), RPM Fusion (http://rpmfusion.org/), or someplace else? No problem.

Now that we've added the mirrors, let's pull down the content. This will take a little while, but subsequent updates won't take nearly as long.

```
cobbler reposync
```

Now, that the repositories are mirrored locally, let's create a cobbler profile that will be able to automatically install from the above repositories and also configure clients to use the new mirror.

```
cobbler profile add --name=f12-i386-test --repos="f12-i386-updates f12-i386-
↪everything" --distro=F12-i386 --kickstart=/etc/cobbler/sample_end.ks
```

Now, any machines installed from this mirror won't have to hit the outside world for any content they may need during install or with yum. They'll ask for content from the cobbler server instead. Cool.

3.7.4 RHN

This is rather experimental, but if you have a provisioning need for fast local installs without hitting an outside server repeatedly (say you have a slow pipe), you can try:

```
cobbler repo add --name=insertnamehere --mirror=rhn://rhn-channel-name
```

That's just the channel-name, no server. This only works on RHEL5+ and you'll need entitlements for the channel in question. You also want a version of yum-utils at least equal to 1.0.4.

3.7.5 Post Install Yum Usage

If you want your installed systems to be automatically configured to use your install server for updates, go into `/etc/cobbler/settings` and set the following:

```
yum_post_install_mirror: 1
```

(Don't do this if the servers can't reach the cobbler server at the value set up in settings or if you're going to move the installed machine to a different network later)

3.7.6 Updates And Cron

As you're mirroring repositories that change (and probably even include some security updates from time to time), putting `cobbler reposync` on crontab would be a good idea. Cobbler reposync will update the content in all of your repositories.

You can disable updating of certain repos that you've already pulled down and don't wish to contact again by toggling the `--keep-updated` flag on the repo. Make sure you reposync them at least once.

Use of the following flags will ensure smoother updates from cron: `cobbler reposync --tries=3 --no-fail`

This will allow Cobbler to keep trucking if one of your mirrors has problems.

3.7.7 Also Apt

Starting, we can also do apt mirroring (see Distribution Support).

```
cobbler repo add --name=foo --mirror=http://url --breed=apt --arch=i386
```

This is useful with Debian distributions (those that have `--breed=debian` in the distro object), see [Distribution Support](Distribution Support]

3.7.8 Saving Space

To eliminate space duplicated between mirrored updates and install trees, run the following command: `cobbler hardlink`

This requires that you have first installed the package 'hardlink'.

3.7.9 To Review

The above steps have set up your cobbler server as a full fledged mirror, not just for install trees (which are imported using `cobbler import` not reposync – read [Using cobbler import](/cobbler/wiki/UsingCobblerImport)), but also for future package installs and updates with yum.

Installation content during anaconda and afterwards will be pulled from your cobbler mirror, not the outside world. You should see faster installs and won't have to worry about whether your client machines have outside internet connectivity.

Cobbler handles all of the yum, reposync, and createrepo magic for you, so you don't have to know how they work. Plus, the kickstarts are automatically aware of the configuration and build themselves out based on what repos are defined. Bottom line: you don't need to know how any of this stuff works. Cool.

If you have questions or want to clear up something in this document, ask on the mailing list or stop by `#cobbler` on IRC.

3.8 File System Information

A typical Cobbler install looks something as follows. Note that in general Cobbler manages its own directories. Editing templates and configuration files is intended. Deleting directories will result in very loud alarms. Please do not ask for help if you decide to delete core directories or move them around.

See *Relocating your installation* if you have space problems.

3.8.1 /var/log/cobbler

All Cobbler logs go here. Cobbler does not dump to `/var/log/messages`, though other system services relating to netbooting do.

3.8.2 /var/www/cobbler

This is a Cobbler owned and managed directory for serving up various content that we need to serve up via http. Further selected details are below. As tempting as it is to self-garden this directory, do not do so. Manage it using the "cobbler" command or the Cobbler web app.

/var/www/cobbler/web/

Here is where the `mod_python` web interface and supporting service scripts live for Cobbler pre 2.0

/var/www/cobbler/webui/

Here is where content for the (pre 2.0) webapp lives that is not a template. Web templates for all versions live in `/usr/share/cobbler`.

/var/www/cobbler/aux/

This is used to serve up certain scripts to anaconda, such as anamon (See *Anaconda Monitoring* for more information on anamon).

/var/www/cobbler/svc/

This is where the `mod_wsgi` script for Cobbler lives.

/var/www/cobbler/images/

Kernel and initrd files are copied/symlinked here for usage by koan.

/var/www/cobbler/ks_mirror/

Install trees are copied here.

/var/www/cobbler/repo_mirror/

Cobbler repo objects (i.e. yum, apt-mirror) are copied here.

3.8.3 /var/lib/cobbler/

This is the main data directory for Cobbler. See individual descriptions of subdirectories below.

/var/lib/cobbler/config/

Here Cobbler stores configuration files that it creates when you make or edit Cobbler objects. If you are using `serializer_catalog` in `modules.conf`, these will exist in various "`.d`" directories under this main directory.

/var/lib/cobbler/backups/

This is a backup of the config directory created on RPM upgrades. The configuration format is intended to be forward compatible (i.e. upgrades without user intervention are supported) though this file is kept around in case something goes wrong during an install (though it never should, it never hurts to be safe).

/var/lib/cobbler/kickstarts/

This is where Cobbler's shipped kickstart templates are stored. You may also keep yours here if you like. If you want to edit kickstarts in the web application this is the recommended place for them. Though other distributions may have templates that are not explicitly 'kickstarts', we also keep them here.

/var/lib/cobbler/snippets/

This is where Cobbler keeps snippet files, which are pieces of text that can be reused between multiple kickstarts.

/var/lib/cobbler/triggers/

Various user-scripts to extend Cobbler to perform certain actions can be dropped into subdirectories of this directory. See the *Triggers* section for more information.

3.8.4 /usr/share/cobbler/web

This is where the cobbler-web package (for Cobbler 2.0 and later) lives. It is a Django app.

3.8.5 /etc/cobbler/

- **cobbler.conf** - Cobbler's most important config file. Self-explanatory with comments, in YAML format.
- **modules.conf** - auxilliary config file. controls Cobbler security, and what DHCP/DNS engine is attached, see *Modules* for developer-level details, and also Security Overview. This file is in an INI-style format that can be read by the ConfigParser class.
- **users.digest** - if using the digest authorization module this is where your web app username/passwords live. Refer to the *Web Interface* section for more info.

/etc/cobbler/power

Here we keep the templates for the various power management modules Cobbler supports. Please refer to the *Power Management* section for more details on configuring power management features.

/etc/cobbler/pxe

Various templates related to netboot installation, not neccessarily "pxe".

/etc/cobbler/zone_templates

If the chosen DNS management module for DNS is BIND, this directory is where templates for each zone file live. dnsmasq does not use this directory.

/etc/cobbler/reporting

Templates for various reporting related functions of Cobbler, most notably the new system email feature in Cobbler 1.5 and later.

3.8.6 /usr/lib/python${VERSION}/site-packages/cobbler/

The source code to Cobbler lives here. If you have multiple versions of python installed, make sure Cobbler is in the site-packages directory for the correct python version (you can use symlinks to make it available to multiple versions).

/usr/lib/python${VERSION}/site-packages/cobbler/modules/

This is a directory where modules can be dropped to extend Cobbler without modifying the core. See *Modules* for more information.

CHAPTER 4

Advanced

4.1 Advanced Networking

4.1.1 General

This page details some of the networking tips and tricks in more detail, regarding what you can set on system records to set up networking, without having to know a lot about kickstart/Anaconda.

These features include:

- Arbitrary NIC naming (the interface is matched to a physical device using it's MAC address)
- Configuring DNS nameserver addresses
- Setting up NIC bonding
- Defining for static routes
- Support for VLANs

If you want to use any of these features, it's highly recommended to add the MAC addresses for the interfaces you're using to Cobbler for each system.

Arbitrary NIC naming

You can give your network interface (almost) any name you like.

```
```

$ cobbler system edit –name=foo1.bar.local –interface=mgmt –mac=AA:BB:CC:DD:EE:F0 $ cobbler system edit –name=foo1.bar.local –interface=dmz –mac=AA:BB:CC:DD:EE:F1

The default interface is named eth0, but you don't have to call it that.

Note that you can't name your interface after a kernel module you're using. For example: if a NIC is called 'drbd', the module `drbd.ko` would stop working. This is due to an "alias" line in `/etc/modprobe.conf`.

Name Servers

For static systems, the `--name-servers` parameter can be used to specify a list of name servers to assign to the systems.

```
$ cobbler system edit --name=foo --interface=eth0 --mac=AA:BB:CC::DD:EE:FF --
↪static=1 --name-servers="<ip1> <ip2>"
```

Static routes

You can define static routes for a particular interface to use with `--static-routes`. The format of a static route is: `network/CIDR:gateway`

So, for example to route the 192.168.1.0/24 network through 192.168.1.254:

```
$ cobbler system edit --name=foo --interface=eth0 --static-routes="192.168.1.
↪0/24:192.168.1.254"
```

As with all lists in cobbler, the `--static-routes` list is space-separated so you can specify multiple static routes if needed.

Kickstart Notes

Three different networking *Snippets* must be present in your kickstart files for this to work:

```
pre_install_network_config
network_config
post_install_network_config
```

The default kickstart templates (`/var/lib/cobbler/kickstart/sample*.ks`) have these installed by default so they work out of the box. Please use those files as a reference as to where to correctly include the `$SNIPPET` definitions.

4.1.2 Bonding

Bonding is also known as trunking, or teaming. Different vendors use different names. It's used to join multiple physical interfaces to one logical interface, for redundancy and/or performance.

You can set up a bond, to join interfaces eth0 and eth1 to a failover (active-backup) interface bond0 as follows:

```
$ cobbler system edit --name=foo --interface=eth0 --mac=AA:BB:CC:DD:EE:F0 --
↪interface-type=bond_slave --interface-master=bond0
$ cobbler system edit --name=foo --interface=eth1 --mac=AA:BB:CC:DD:EE:F1 --
↪interface-type=bond_slave --interface-master=bond0
$ cobbler system edit --name=foo --interface=bond0 --interface-type=bond --
↪bonding-opts="miimon=100 mode=1" --ip-address=192.168.1.100 --netmask=255.
↪255.255.0
```

You can specify any bonding options you would like, so please read the kernel documentation if you are unfamiliar with the various bonding modes Linux can support.

Notes About Bonding Syntax

The methodology to create bonds was changed in 2.2.x with the introduction of bridged interface support. The **–bonding** and **–bonding-master** options have since been deprecated and are now an alias to **–interface-type** and **–interface-master**, respectively.

Likewise, the master/slave options have been deprecated in favor of bond/bond_slave. Cobbler will continue to read system objects that have those fields set, but when the object is edited and saved back to disk they will be converted to the new format transparently.

4.1.3 VLANs

You can now add VLAN tags to interfaces from Cobbler. In this case we have two VLANs on eth0: 10 and 20. The default VLAN (untagged traffic) is not used:

```
$ cobbler system edit --name=foo3.bar.local --interface=eth0 --
↪mac=AA:BB:CC:DD:EE:F0 --static=1
$ cobbler system edit --name=foo3.bar.local --interface=eth0.10 --static=1 --
↪ip-address=10.0.10.5 --subnet=255.255.255.0
$ cobbler system edit --name=foo3.bar.local --interface=eth0.20 --static=1 --
↪ip-address=10.0.20.5 --subnet=255.255.255.0
```

Note: You must install the vconfig package during the build process for this to work in the %post section of your build.

4.1.4 Bridging

A bridge is a way to connect two Ethernet segments together in a protocol independent way. Packets are forwarded based on Ethernet address, rather than IP address (like a router). Since forwarding is done at Layer 2, all protocols can go transparently through a bridge. (reference (https://wiki.linuxfoundation.org/networking/bridge)).

You can create a bridge in cobbler in the following way:

```
$ cobbler system edit --name=foo --interface=eth0 --mac=AA:BB:CC:DD:EE:F0 --
↪interface-type=bridge_slave --interface-master=br0
$ cobbler system edit --name=foo --interface=eth1 --mac=AA:BB:CC:DD:EE:F1 --
↪interface-type=bridge_slave --interface-master=br0
$ cobbler system edit --name=foo --interface=br0 --interface-type=bridge --
↪bridge-opts="stp=no" --ip-address=192.168.1.100 --netmask=255.255.255.0
```

You can specify any bridging options you would like, so please read the brctl manpage for details if you are unfamiliar with bridging.

Note: You must install the `bridge-utils` package during the build process for this to work in the `%post` section of your build.

4.1.5 Bonded Bridging

Some situations, such as virtualization hosts, require more redundancy in their bridging setups. In this case, 2.8.0 introduced a new interface type - the `bonded_bridge_slave`. This is an interface that is a bond master to one or more physical interfaces, and is itself a bridged slave interface.

You can create a `bonded_bridge_slave` in cobbler in the following way:

```
$ cobbler system edit --name=foo --interface=eth0 --mac=AA:BB:CC:DD:EE:F0 \
                      --interface-type=bond_slave --interface-master=bond0
$ cobbler system edit --name=foo --interface=eth1 --mac=AA:BB:CC:DD:EE:F1 \
                      --interface-type=bond_slave --interface-master=bond0
$ cobbler system edit --name=foo --interface=bond0 --interface-type=bonded_
↪bridge_slave \
                      --bonding-opts="miimon=100 mode=1" --interface-
↪master=br0
$ cobbler system edit --name=foo --interface=br0 --interface-type=bridge \
                      --bridge-opts="stp=no" --ip-address=192.168.1.100 \
                      --netmask=255.255.255.0 --static=1
```

Note: Please reference the *Bonding* and *Bridging* sections for requirements specific to each of these interface types.

4.2 SELinux

SELinux policies are typically provided by the upstream distribution (Fedora, Ubuntu, etc.). As new features are added to cobbler (and we do add new features frequently), those policies may become out-of-date leading to AVC denials and other problems. If you wish to run SELinux on your cobbler system, we expect you to know how to write policy and resolve AVCs.

Below are some of the more common issues you may run into with this release.

4.2.1 ProtocolError: <ProtocolError for x.x.x.x:80/cobbler_api: 503 Service Temporarily Unavailable>

If you see this when you run `cobbler check` or any other cobbler command, it means SELinux is blocking httpd from talking with cobblerd. The command to fix this is:

```
$ sudo setsebool -P httpd_can_network_connect true
```

4.2.2 Fedora 16 / RHEL6 / CentOS6 - Python MemoryError

When starting cobblerd for the first time (or after upgrading to 2.2.x), you may see a stack trace like the following:

```
Starting cobbler daemon: Traceback (most recent call last):
File "/usr/bin/cobblerd", line 76, in main
api = cobbler_api.BootAPI(is_cobblerd=True)
File "/usr/lib/python2.6/site-packages/cobbler/api.py", line 127, in init
module_loader.load_modules()
File "/usr/lib/python2.6/site-packages/cobbler/module_loader.py", line 62, in␣
↪load_modules
blip = import("modules.%s" % ( modname), globals(), locals(), [modname])
File "/usr/lib/python2.6/site-packages/cobbler/modules/authn_pam.py", line 53,
↪ in
from ctypes import CDLL, POINTER, Structure, CFUNCTYPE, cast, pointer, sizeof
File "/usr/lib64/python2.6/ctypes/init.py", line 546, in
CFUNCTYPE(c_int)(lambda: None)
MemoryError
```

This error is caused by SELinux blocking python ctypes. To resolve this, you can use audit2allow to enable the execution of temp files or you can remove the authn_pam.py module from the site-packages/cobbler/modules directory (as long as you're not using PAM authentication for the Web UI).

4.3 Configuration Management

The initial provisioning of client systems with cobbler is just one component of their management. We also need to consider how to continue to manage them using a configuration management system (CMS). Cobbler can help you provision and introduce a CMS onto your client systems.

One option is cobbler's own lightweight CMS. For that, see the document Built in configuration management.

Here we discuss the other option: deploying a CMS such as puppet, cfengine, bcfg2, Chef, etc.

Cobbler doesn't force you to chose a particular CMS (or to use one at all), though it helps if you do some things to link cobbler's profiles with the "profiles" of the CMS. This, in general, makes management of both a lot easier.

Note that there are two independent "variables" here: the possible client operating systems and the possible CMSes. We don't attempt to cover all details of all combinations; rather we illustrate the principles and give

a small number of illustrative examples of particular OS/CMS combinations. Currently cobbler has better support for Redhat-based OSes and for Puppet so the current examples tend to deal with this combination.

4.3.1 Background considerations

Machine lifecyle

A typical computer has a lifecycle something like:

- installation
- initial configuration
- ongoing configuration and maintenance
- decommissioning

Typically installation happens once. Likewise, the initial configuration happens once, usually shortly after installation. By contrast ongoing configuration evolves over an extended period, perhaps of several years. Sometimes part of that ongoing configuration may involve re-installing an OS from scratch. We can regard this as repeating the earlier phase.

We need not consider decommissioning here.

Installation clearly belongs (in our context) to Cobbler. In a complementary manner, ongoing configuration clearly belongs to the CMS. But what about initial configuration?

Some sites consider their initial configuration as the final phase of installation: in our context, that would put it at the back end of Cobbler, and potentially add significant configuration-based complication to the installation-based Cobbler set-up.

But it is worth considering initial configuration as the first step of ongoing configuration: in our context that would put it as part of the CMS, and keep the Cobbler set-up simple and uncluttered.

Local package repositories

Give consideration to:

- local mirrors of OS repositories
- local repository of local packages
- local repository of pick-and-choose external packages

In particular consider having the packages for your chosen CMS in one of the latter.

Package management

Some sites set up Cobbler always to deploy just a minimal subset of packages, then use the CMS to install many others in a large-scale fashion. Other sites may set up Cobbler to deploy tailored sets of packages to different types of machines, then use the CMS to do relatively small-scale fine-tuning of that.

4.3.2 General scheme

We need to consider getting Cobbler to install and automatically invoke the CMS software.

Set up Cobbler to include a package repository that contains your chosen CMS:

```
cobbler repo add ...
```

Then (illustrating a Redhat/Puppet combination) set up the kickstart file to say something like:

```
%packages
puppet

%post
/sbin/chkconfig --add puppet
```

The detail may need to be more substantial, requiring some other associated local packages, files and configuration. You may wish to manage this through *Snippets*.

4.3.3 Built-In Configuration Management

Cobbler is not just an installation server, it can also enable two different types of ongoing configuration management system (CMS):

- integration with an established external CMS such as cfengine3 (https://cfengine.com/) or Puppet (https://puppet.com/#), discussed *elsewhere*;
- its own, much simpler, lighter-weight, internal CMS, discussed here.

Setting up

Cobbler's internal CMS is focused around packages and templated configuration files, and installing these on client systems.

This all works using the same Cheetah-powered (https://cheetahtemplate.org/) templating engine used in *Kickstart Templating*, so once you learn about the power of treating your distribution answer files as templates, you can use the same templating to drive your CMS configuration files.

For example:

```
cobbler profile edit --name=webserver \
  --template-files=/srv/cobbler/x.template=/etc/foo.conf
```

A client system installed via the above profile will gain a file `/etc/foo.conf` which is the result of rendering the template given by `/srv/cobbler/x.template`. Multiple files may be specified; each `template=destination` pair should be placed in a space-separated list enclosed in quotes:

```
--template-files="srv/cobbler/x.template=/etc/xfile.conf srv/cobbler/y.
↪template=/etc/yfile.conf"
```

Template files

Because the template files will be parsed by the Cheetah parser, they must conform to the guidelines described in *Kickstart Templating*. This is particularly important when the file is generated outside a Cheetah environment. Look for, and act on, Cheetah 'ParseError' errors in the Cobbler logs.

Template files follows general Cheetah syntax, so can include Cheetah variables. Any variables you define anywhere in the cobbler object hierarchy (distros, profiles, and systems) are available to your templates. To see all the variables available, use the command:

```
cobbler profile dumpvars --name=webserver
```

Cobbler snippets and other advanced features can also be employed.

Ongoing maintenance

Koan can pull down files to keep a system updated with the latest templates and variables:

```
koan --server=cobbler.example.org --profile=foo --update-files
```

You could also use `--server=bar` to retrieve a more specific set of templating.(???) Koan can also autodetect the server if the MAC address is registered.

Further uses

This Cobbler/Cheetah templating system can serve up templates via the magic URLs (see "Leveraging Mod Python" below). To do this ensure that the destination path given to any `--template-files` element is relative, not absolute; then Cobbler and koan won't download those files.

For example, in:

```
cobbler profile edit --name=foo \
  --template-files="/srv/templates/a.src=/etc/foo/a.conf /srv/templates/b.
↪src=1"
```

cobbler and koan would automatically download the rendered `a.src` to replace the file `/etc/foo/a.conf`, but the `b.src` file would not be downloaded to anything because the destination pathname `1` is not absolute.

This technique enables using the Cobbler/Cheetah templating system to build things that other systems can fetch and use, for instance, BIOS config files for usage from a live environment.

Leveraging Mod Python

All template files are generated dynamically at run-time. If a change is made to a template, a `--ks-meta` variable or some other variable in cobbler, the result of template rendering will be different on subsequent runs. This is covered in more depth in the Developer Documentation (https://github.com/cobbler/cobbler/wiki).

Possible future developments

- Serving and running scripts via `--update-files` (probably staging them through `/var/spool/koan`).
- Auto-detection of the server name if `--ip` is registered.

4.3.4 Puppet Integration

This example is relatively advanced, involving Cobbler "mgmt-classes" to control different types of initial configuration. But if instead you opt to put most of the initial configuration into the Puppet CMS rather than here, then things could be simpler.

Keeping Class Mappings In Cobbler

First, we assign management classes to distro, profile, or system objects.

```
cobbler distro edit --name=distro1 --mgmt-classes="distro1"
cobbler profile add --name=webserver --distro=distro1 --mgmt-classes=
↪"webserver likes_llamas" --kickstart=/etc/cobbler/my.ks
cobbler system edit --name=system --profile=webserver --mgmt-classes="orange"␣
↪--dns-name=system.example.org
```

For Puppet, the `--dns-name` (shown above) must be set because this is what puppet will be sending to cobbler and is how we find the system. Puppet doesn't know about the name of the system object in cobbler. To play it safe you probably want to use the FQDN here (which is also what you want if you were using Cobbler to manage your DNS, which you don't have to be doing).

External Nodes

For more documentation on Puppet's external nodes feature, see https://puppet.com/docs

Cobbler provides one, so configure puppet to use `/usr/bin/cobbler-ext-nodes`:

```
[main]
external_nodes = /usr/bin/cobbler-ext-nodes
```

Note: If you are using puppet 0.24 or later then you will want to also add the following to your configuration file `node_terminus = exec`

You may wonder what this does. This is just a very simple script that grabs the data at the following URL, which is a URL that always returns a YAML document in the way that Puppet expects it to be returned. This file contains all the parameters and classes that are to be assigned to the node in question. The magic URL being visited is powered by Cobbler: `http://cobbler/cblr/svc/op/puppet/hostname/foo`

And this will return data such as:

```
---
classes:
    - distro1
    - webserver
    - likes_llamas
    - orange
parameters:
    tree: 'http://.../x86_64/tree'
```

Where do the parameters come from? Everything that cobbler tracks in `--ks-meta` is also a parameter. This way you can easily add parameters as easily as you can add classes, and keep things all organized in one place.

What if you have global parameters or classes to add? No problem. You can also add more classes by editing the following fields in `/etc/cobbler/settings`:

```
mgmt_classes: []
mgmt_parameters:
   from_cobbler: 1
```

Alternate External Nodes Script

Attached at puppet_node.py is an alternate external node script that fills in the nodes with items from a manifests repository (at `/etc/puppet/manifests/`) and networking information from cobbler. It is configured like the above from the puppet side, and then looks for `/etc/puppet/external_node.yaml` for cobbler side configuration.

The configuration is as follows.

```
base: /etc/puppet/manifests/nodes
cobbler: <%= cobbler_host %>
no_yaml: puppet::noyaml
no_cobbler: network::nocobbler
bad_yaml: puppet::badyaml
unmanaged: network::unmanaged
```

The output for network information will be in the form of a pseudo data structure that allows puppet to split it apart and create the network interfaces on the node being managed.

4.3.5 Func Integration

> **Warning:** This feature has been deprecated and will not be available in Cobbler 3.0.

Func is a neat tool, (which, in full disclosure, Michael had a part in creating).

Integration

Cobbler makes it even easier to deploy Func though. We have two settings in `/etc/cobbler/settings`:

```
func_master: overlord.example.org
func_auto_setup: 1
```

This will make sure the right packages are in packages for each kickstart and the right bits are automatically in %post to set it up... so a new user can set up a cobbler server, set up a func overlord, and automatically have all their new kickstarts configurable to point at that overlord.

This will be available in all the sample kickstart files, but will be off by default. To enable this feature all you need to do then is set up

How This Is Implemented

This is all powered by cobbler's *Kickstart Templating* and *Snippets* feature, with two snippets that ship stock in `/var/lib/cobbler/snippets`

```
%packages
koan
...
$func_install_if_enabled

%post
...
SNIPPET:func_register_if_enabled
```

If curious you can read the implementations in `/var/lib/cobbler/snippets` and these are of course controlled by the aforemented values in settings.

The `func_register_if_enabled` snippet is pretty basic.

It configures func to point to the correct certmaster to get certificates and enables the service. When the node boots into the OS it will request the certificate (see note on autosigning below) and func is now operational. If there are problems, see `/var/log/func` and `/var/log/certmaster` for debugging info (or other resources and information on the Func Wiki page).

Notes about Func Autosigning

This may work better for you if you are using Func autosigning, otherwise the administrator will need to use `certmaster-ca --sign hostname` (see also `certmaster-ca --list`) to deal with machines.

Not using autosigning is good if you don't trust all the hosts you are provisioning and don't want to enslave unwanted machines.

Either choice is ok, just be aware of the manual steps required if you don't enable it, or the implications if you do.

Package Hookup

If you are not already using Cobbler to mirror package content, you are going to want to, so that you can make the func packages available to your systems – they are not part of the main install "tree".

Thankfully Cobbler makes this very simple – see Manage Yum Repos for details

for Fedora

Func is part of the package set for Fedora, but you need to mirror the "Everything" repo to get at it. Therefore you will want to mirror "Everything" and make it available to your cobbler profiles so you can effectively put func on your installed machines. You will also want to mirror "updates" to make sure you get the latest func.

An easy way to mirror these with cobbler is just:

```
cobbler repo add --name=f10-i386-updates --mirror=http://download.fedora.
↪redhat.com/pub/fedora/linux/updates/10/i386/
cobbler repo add --name=f10-i386-everything --mirror=http://download.fedora.
↪redhat.com/pub/fedora/linux/releases/10/Everything/i386/os/Packages/
```

Then you need to make sure that every one of your Fedora profiles is set up to use the appropriate repos:

```
cobbler profile edit --name=f10-profile-name-goes-here --repos="f10-i386-
↪updates f10-i386-everything"
```

And then you would probably want to put `cobbler reposync` on cron so you keep installing the latest func, not an older func.

for Enterprise Linux 4 and 5

As with Fedora, you'll need to configure your systems as above to get func onto them, and that is not included as part of the Func integration process. RHEL 5 uses yum, so it can follow similar instructions as above. That's very simple. In those cases you will just want to mirror the repositories for EPEL:

```
cobbler repo add --name=el-5-i386-epel --mirror=http://download.fedora.redhat.
↪com/pub/epel/5/i386
cobbler repo add --name=el-5-i386-epel-testing --mirror=http://download.
↪fedora.redhat.com/pub/epel/testing/5/i386
```

Of course in the above you would want to substitute '4' for '5' if neccessary and also 'i386' for 'x86_64' if neccessary. You will probably want to mirror multiples of the above. Cobbler doesn't care, just go ahead and do it. If you have space concerns, as discussed on Manage Yum Repos you can use the `--rpm-list` parameter to do partial yum mirroring.

Once you do this, you will need to make sure your EL profiles (for those that support yum, i.e. the EL 5 and later ones) know about the repos and attach to them automatically:

```
cobbler profile edit --name=el5-profile-name-goes-here --repos="el-5-i386-
↪epel el-5-i386-epel-testing"
```

Another simple option is to just put the func RPMs on a webserver somewhere and wget them from the installer so they are available at install time, you would do this as the very first step in post.

```
%post
wget http://myserver.example.org/func-version.rpm -O /tmp/func.rpm
rpm -i /tmp/func.rpm
```

Func Questions

See `#func` on irc.freenode.net and func-list@redhat.com

4.3.6 Conclusion

Hopefully this should get you started in linking up your provisioning configuration with your CMS implementation. The examples provided are for Puppet, but we can (in the future) presumably extend `--mgmt-classes` to work with other tools... just let us know what you are interested in, or perhaps take a shot at creating a patch for it.

4.4 Extending cobbler

This section covers methods to extend the functionality of Cobbler through the use of *Triggers* and *Modules*, as well as through extensions to the Cheetah templating system.

4.4.1 Triggers

About

Cobbler triggers provide a way to tie user-defined actions to certain cobbler commands – for instance, to provide additional logging, integration with apps like Puppet or cfengine, set up SSH keys, tieing in with a DNS server configuration script, or for some other purpose.

Cobbler Triggers should be Python modules written using the low-level Python API for maximum speed, but could also be simple executable shell scripts.

As a general rule, if you need access to Cobbler's object data from a trigger, you need to write the trigger as a module. Also never invoke cobbler from a trigger, or use Cobbler XMLRPC from a trigger. Essentially, cobbler triggers can be thought of as plugins into cobbler, though they are not essentially plugins per se.

Trigger Names (for Old-Style Triggers)

Cobbler script-based triggers are scripts installed in the following locations, and must be made `chmod +x`.

- /var/lib/cobbler/triggers/add/system/pre/*
- /var/lib/cobbler/triggers/add/system/post/*
- /var/lib/cobbler/triggers/add/profile/pre/*
- /var/lib/cobbler/triggers/add/profile/post/*
- /var/lib/cobbler/triggers/add/distro/pre/*
- /var/lib/cobbler/triggers/add/distro/post/*
- /var/lib/cobbler/triggers/add/repo/pre/*
- /var/lib/cobbler/triggers/add/repo/post/*
- /var/lib/cobbler/triggers/sync/pre/*
- /var/lib/cobbler/triggers/sync/post/*
- /var/lib/cobbler/triggers/install/pre/*
- /var/lib/cobbler/triggers/install/post/*

And the same as the above replacing "add" with "remove".

Pre-triggers are capable of failing an operation if they return anything other than 0. They are to be thought of as "validation" filters. Post-triggers cannot fail an operation and are to be thought of notifications.

We may add additional types as time goes on.

Pure Python Triggers

As mentioned earlier, triggers can be written in pure python, and many of these kinds of triggers ship with cobbler as stock.

Look in your site-packages/cobbler/modules directory and cat "install_post_report.py" for an example trigger that sends email when a system gets done installing.

Notice how the trigger has a register method with a path that matches with the shell patterns above. That's how we know what type each trigger is.

You will see the path used in the trigger corresponds with the path where it would exist if it was a script – this is how we know what type of trigger the module is providing.

The Simplest Trigger Possible

1. Create `/var/lib/cobbler/triggers/add/system/post/test.sh`.
2. chmod +x the file.

```
#!/bin/bash
echo "Hi, my name is $1 and I'm a newly added system"
```

However that's not very interesting as all you get are the names passed across. For triggers to be the most powerful, they should take advantage of the Cobbler API – which means writing them as a Python module.

Performance Note

If you have a very large number of systems, using the Cobbler API from scripts with old style (non-Python modules, just scripts in `/var/lib/cobbler/triggers`) is a very very bad idea. The reason for this is that the cobbler API brings the cobbler engine up with it, and since it's a seperate process, you have to wait for that to load. If you invoke 3000 triggers editing 3000 objects, you can see where this would get slow pretty quickly. However, if you write a modular trigger (see above) this suffers no performance penalties – it's crazy fast and you experience no problems.

Permissions

The `/var/lib/cobbler/triggers` directory is only writeable by root (and are executed by cobbler on a regular basis). For security reasons do not loosen these permissions.

Example trigger for resetting Cfengine keys

Here is an example where cobbler and cfengine are running on two different machines and xmlrpc is used to communicate between the hosts.

Note that this uses the Cobbler API so it's somewhat inefficient – it should be converted to a Python module-based trigger. If it were in a pure Python modular trigger, it would fly.

On the cobbler box: `/var/lib/cobbler/triggers/install/post/clientkeys.py`

```
#!/usr/bin/python
import socket
import xmlrpclib
import sys
from cobbler import api
cobbler_api = api.BootAPI()
systems = cobbler_api.systems()
box = systems.find(sys.argv[2])
server = xmlrpclib.ServerProxy("http://cfengine:9000")
server.update(box.get_ip_address())
```

On the cfengine box, we run a daemon that does the following (along with a few steps to update our ssh_known_hosts file):

```
#!/usr/bin/python
import SimpleXMLRPCServer
import os
class Keys(object):
     def update(self, ip):
         try:
            os.unlink('/var/cfengine/ppkeys/root-%s.pub' % ip)
        except OSError:
            pass
keys = Keys()
server = SimpleXMLRPCServer.SimpleXMLRPCServer(("cfengine",9000))
```

(continues on next page)

(continued from previous page)

```
server.register_instance(keys)
server.serve_forever()
```

See Also

- Post by Ithiriel: Writing triggers (https://www.ithiriel.com/content/2010/03/29/writing-install-triggers-cobbler)

4.4.2 Modules

Certain cobbler features can be user extended (in Python) by Cobbler users.

These features include storage of data (serialization), authorization, and authentication. Over time, this list of module types will grow to support more options. *Triggers* are basically modules.

See Also

- *Security Overview*
- The cobbler command line itself (it's implemented in cobbler modules so it's easy to add new commands)

Python Files And modules.conf

To create a module, add a python file in `/usr/lib/python$version/site-packages/cobbler/modules`. Then, in the appropriate part of `/etc/cobbler/modules.conf`, reference the name of your module so cobbler knows that you want to activate the module.

(*Triggers* that are python modules, as well as CLI python modules don't need to be listed in this file, they are auto-loaded)

An example from the serializers is:

```
[serializers]
settings = serializer_catalog
```

The format of `/etc/cobbler/modules.conf` is that of Python's ConfigParser module.

A setup file consists of sections, lead by a "[section]" header, and followed by "name: value" entries, with continuations and such in the style of RFC 822.

Each module, regardless of it's nature, must have the following function that returns the type of module (as a string) on an acceptable load (when the module can be loaded) or raises an exception otherwise.

The trivial case for a cli module is:

```
def register():
    return "cli"
```

Other than that, modules do not have a particular API signature – they are "Duck Typed" based on how they are employed. When starting a new module, look at other modules of the same type to see what functions they possess.

4.4.3 Extending Cheetah

Cobbler uses Cheetah for it's templating system, it also wants to support other choices and may in the future support others.

It is possible to add new functions to the templating engine, much like snippets, that provide the ability to do macro-based things in the template. If you are new to Cheetah, see the documentation at https://cheetahtemplate.org/users_guide/index.html and pay special attention to the `#def` directive.

To create new functions, add your Cheetah code to `/etc/cobbler/cheetah_macros`. This file will be sourced in all Cheetah templates automatically, making it possible to write custom functions and use them from this file.

You will need to restart cobblerd after changing the macros file.

4.5 Power Management

Cobbler allows for linking your power management systems with cobbler, making it very easy to make changes to your systems when you want to reinstall them, or just use it to remember what the power management settings for all of your systems are. For instance, you can just change what profile they should run and flip their power states to begin the reinstall!

4.5.1 Fence Agents

Cobbler relies on fencing agents, provided by the 'cman' package for some distributions or 'fence-agents' for others. These scripts are installed in the `/usr/sbin` directories. Cobbler will automatically find any files in that directory named `fence_*` and allow them to be used for power management.

Note: Some distros may place the fencing agents in `/sbin` - this is currently a known bug. To work around this for now, symlink the `/sbin/fence_*` scripts you wish to use to `/usr/sbin` so cobbler can find them. This will be fixed in a future version.

4.5.2 Changes From Older Versions

Cobbler versions prior to 2.2.3-2 used templates stored in `/etc/cobbler/power` to generate commands that were run as shell commands. This was changed in 2.2.3-2 to use the fencing agents ability to instead read the parameters from STDIN. This is safer, as no passwords are shown in plaintext command line options, nor can a malformed variable be used to inject improper shell commands during the fencing agent execution.

New Power Templates

By default, the following options are passed in to the fencing agent's STDIN:

```
action=$power_mode
login=$power_user
passwd=$power_pass
ipaddr=$power_address
port=$power_id
```

The variables above correspond to the `--power-*` options available when adding/editing a system (or via the "Power Management" tab in the Web UI). If you wish to add aditional options, you can create a template in `/etc/cobbler/power` named `fence_<name>`.template, where name is the fencing agent you wish to use.

Any additional options should be added one per line, as described in the fencing agents man page. Additional variables can be used if they are set in `--ksmeta`.

Custom Fencing Agents

If you would like to use a custom fencing agent not provided by your distribution, you can do so easily by placing it in the `/usr/sbin` directory and name it `fence_<mytype>`. Just make sure that your custom program reads its options from STDIN, as noted above.

4.5.3 Defaults

If `--power-user` and `--power-pass` are left blank, the values of `default_power_user` and `default_power_pass` will be loaded from cobblerd's environment at the time of usage.

`--power-type` also has a default value in `/etc/cobbler/settings`, initially set to "ipmilan".

4.5.4 Important: Security Implications

Storing the power control usernames and passwords in Cobbler means that information is essentially public (this data is available via XMLRPC without access control), therefore you will want to control what machines have network access to contact the power management devices if you use this feature (such as /only/ the cobbler machine, and then control who has local access to the cobbler machine). Also do not reuse important passwords for your power management devices. If this concerns you, you can still use this feature, just don't store the username/password in Cobbler for your power management devices.

If you are not going to to store power control passwords in Cobbler, leave the username and password fields blank. Cobbler will first try to source them from it's environment using the `COBBLER_POWER_USER` and `COBBLER_POWER_PASS` variables.

This may also be too insecure for some, so in this case, don't set these, and supply `--power-user` and `--power-pass` when running commands like `cobbler system poweron` and `cobbler system poweroff`. The values used on the command line are always used, regardless of the value stored in Cobbler or the environment, if so provided.

```
$ cobbler system poweron --name=foo --power-user=X --power-pass=Y
```

Be advised of current limitations in storing passwords, make your choices accordingly and in relation to the ease-of-use that you need, and secure your networks appropriately.

4.5.5 Sample Use

Configuring Power Options on a System

You have a DRAC based blade:

```
$ cobbler system edit --name=foo --power-type=drac --power-address=blade-mgmt.
↪example.org --power-user=Administrator --power-pass=PASSWORD --power-
↪id=blade7
```

You have an IPMI based system:

```
$ cobbler system edit --name=foo --power-type=ipmilan --power-address=foo-
↪mgmt.example.org --power-user=Administrator --power-pass=PASSWORD
```

You have a IBM HMC managed system:

```
$ cobbler system edit --name=foo --power-type=lpar --power-address=ibm-hmc.
↪example.org --power-user=hscroot --power-pass=PASSWORD --power-
↪id=system:partition
```

Note: The `--power-id` option is used to indicate both the managed system name **and** a logical partition name. Since an IBM HMC is responsible for managing more than one system, you must supply the managed system name and logical partition name separated by a colon (':') in the `--power-id` command-line option.

You have an IBM Bladecenter:

```
$ cobbler system edit --name=foo --power-type=bladecenter --power-
↪address=blademm.example.org --power-user=USERID --power-pass=PASSW0RD --
↪power-id=6
```

Note: The `--power-id` option is used to specify what slot your blade is connected.

Powering Off A System

```
$ cobbler system poweroff --name=foo
```

Powering On A System

```
$ cobbler system poweron --name=foo
```

If `--netboot-enabled` is not set to false, the system could potentially reinstall itself if PXE has been configured, so make sure to disable that option when using power management features.

Rebooting A System

```
$ cobbler system reboot --name=foo
```

Since not all power management systems support reboot, this is a "power off, sleep for 1 second, and power on" operation.

4.6 Alternative template formats

The default templating engine currently is Cheetah, as of cobbler 2.4.0 support for the Jinja2 templating engine has been added.

The default template type to use in the absence of any other detected. If you do not specify the template with `#template=<template_type>` on the first line of your templates/snippets, cobbler will assume try to use the template engine as specified in the settings file to parse the templates.

From `/etc/cobbler/settings`:

```
# Current valid values are: cheetah, jinja2
default_template_type: "cheetah"
```

4.7 Multi-Homes cobbler servers

How do you use Cobbler to provision machines onto multiple different subnets, assuming no DNS tricks can be employed, and where the server to provision the systems has different addresses on both sides of the "fence". Normally I'd hope that you could say "bootserver" as a resolvable hostname is resolved on all sides, but what if you can't have that? Suppose you have no working DNS. Now you need to be able to have different values for the address of the cobbler server.

Each profile and system object can take a `--server` parameter, which will replace the value used for 'server' in the cobbler settings file.

```
cobbler system edit --name=foo --server=server2.example.org
```

This parameter can also be used to support multiple cobbler servers from a centrally managed configuration (though cobbler replicate is better suited to the task.

4.8 Auto registration

You have a lot of machines coming right "off the truck" and you want to plug them in, install them however (perhaps using PXE menus, koan with `--profile`, a live CD, etc) and then have the system mac address recorded in cobbler so you can change things later.

In the simplest of use cases, you could set up a cobbler default profile that did nothing more than serve up a mostly blank kickstart, set up a post install trigger to email your admin, and then all turned on machines would register automagically, and then you could reassign them to other profiles using the cobbler command line tools. (See *Triggers*)

There are then quite a few ways in which you might want to use this.

4.8.1 Warning

Though this feature cannot overwrite system records, it does have the ability to create a LOT of cobbler system records remotely if someone wanted to write a script to do it. It should not be enabled on a public network, so use your judgement. In most cases it's perfectly safe.

For instance, this was originally written for FreeLinuxPC.org, which had a large amount of machines on a private network that was for installation purposes only. On a restricted installation-only subnet this should be fine.

4.8.2 How It Works

A tool called `cobbler-register` is installed as part of the koan package.

There is also a *Snippets* that is in every default kickstart that says "if this feature is enabled in the settings file, and this is a profile based install, register this system automatically in %post to create a system record for this system if it did not already exist.

If you want to run `cobbler-register` over SSH instead that works too.

`cobbler-register` is smart and will set all fields to "netboot disabled" to prevent newly registered systems from being reinstalled unless you ask them to be.

```
cobbler-register [---server=cobbler.example.org] --profile=name [--
↪hostname=override-hostname.example.org]
```

The server will be sourced from the environment variable `COBBLER_SERVER` if installed previously with cobbler, meaning you won't need to provide it in many cases. The hostname can also be used if available.

This registration script will populate all physical interfaces found, including IP, MAC, and netmask info. The user may wish to fill in additional information later by a script or using web interface or the command line – not everything can be autodetected.

4.9 Batch editing

Do you want to apply a change to a lot of cobbler objects at once?

Try using xargs combined with `cobbler list` commands, such as:

```
cobbler profile list | xargs -n1 --replace cobbler profile edit --virt-
↪bridge=xenbr1 --name={}
```

The above example sets the virtual bridge used by every cobbler profile to 'xenbr1'.

You can filter the profile list by sticking a `grep` commmand in there as a pipe before the xargs.

See also *Command Line Search*

4.10 Moving to a new server

4.10.1 About

What if you have Cobbler installed on Box A, and decide it really needs to be on Box B? Without doing everything from scratch again, how would you accomplish the move? (Some of these instructions may also be useful for backing up a cobbler install as well).

4.10.2 Suggested Process

1. Install Cobbler on the new system.
2. on the new box, run `cobbler replicate`. See *Replication* for instructions and make sure you use the right flags to transfer scripts and data.
3. Try installing some systems to make sure everything works like you would expect.

4.11 PXE boot-menu passwords

4.11.1 How to create a PXE boot menu password

There are two different levels of password:

MENU MASTER PASSWD passwd: Sets a master password. This password can be used to boot any menu entry, and is required for the [Tab] and [Esc] keys to work.

MENU PASSWD passwd: (Only valid after a LABEL statement.) Sets a password on this menu entry. "passwd" can be either a cleartext password or a SHA-1 encrypted password; use the included Perl script "sha1pass" to encrypt passwords. (Obviously, if you don't encrypt your passwords they will not be very secure at all.)

If you are using passwords, you want to make sure you also use the settings "NOESCAPE 1", "PROMPT 0", and either set "ALLOWOPTIONS 0" or use a master password (see below.)

If passwd is an empty string, this menu entry can only be unlocked with the master password.

4.11.2 Creating the password hash

If you have sha1pass on your system (you probably don't, but it's supposed to come with syslinux) you can do: `sha1pass mypassword`

If you do _not_ have sha1pass, you can use openssl to create the pasword (the hashes appear to be compatible): `openssl passwd -1 -salt sXiKzkus mypassword`

4.11.3 Files to edit

- for master menu password: `/etc/cobbler/pxe/pxedefault.template`
- for individual entries: `/etc/cobbler/pxe/pxeprofile.template`

4.11.4 Sample usage

In this example, the master menu password will be used for all the entries (because the profile entry is blank). I have not looked into a way to dynamically set a different password based on the profile variables yet.

```
pxedefault.template:

    DEFAULT menu
    PROMPT 0
    MENU TITLE Cobbler | http://github.com/cobbler
    MENU MASTER PASSWD $1$sXiKzkus$haDZ9JpVrRHBznY5OxB82.

    TIMEOUT 200
    TOTALTIMEOUT 6000
    ONTIMEOUT $pxe_timeout_profile

    LABEL local
            MENU LABEL (local)
            MENU DEFAULT
            LOCALBOOT 0

    $pxe_menu_items

    MENU end


pxeprofile.template:

    LABEL $profile_name
            MENU PASSWD
            kernel $kernel_path
            $menu_label
            $append_line
            ipappend 2
```

4.11.5 References

- `/usr/share/doc/syslinux*/syslinux.doc`
- `/usr/share/doc/syslinux*/README.menu`

4.12 Alternative storage backends

4.12.1 General

Cobbler saves object data via serializers implemented as Cobbler *Modules*. This means Cobbler can be extended to support other storage backends for those that want to do it. Today, cobbler ships three such modules alternate backends: MySQL, MongoDB and CouchDB.

The default serializer is **serializer_catalog** which uses JSON in `/var/lib/cobbler/config/<object>` directories, with one file for each object definition. It is very fast, however people with a large number of systems can still experience slowness, especially if cobbler lives on a disk partition that is slow or heavily utilized. Users with such setups should ensure `/var/lib/cobbler` is mounted on a dedicated disk that offers higher performance (15K SAS or a SAN LUN for example).

An older legacy serializer, "serializer_yaml" is deprecated and is only around to support older installs that have not yet upgraded to serializer_catalog by changing the serializer values in `/etc/cobbler/modules.conf` and restarting cobblerd.

Details

Here's what the relavant parts of modules.conf look like:

```
[serializers]
settings = serializer_catalog
distro = serializer_catalog
profile = serializer_catalog
system = serializer_catalog
repo = serializer_catalog
etc...
```

Note: Be sure to add a line for every object type supported in your version of cobbler. Read the *Cobbler Primitives* section for more details.

Suppose, however, that you (just to be contrary), want to save everything in Marshalled XML because you liked angle brackets a whole lot (we don't!). Easy enough, just write a new serializer module that did this and then could change the file to:

```
[serializers]
settings = serializer_catalog
distro = serializer_xml
```

(continues on next page)

(continued from previous page)

```
profile = serializer_xml
system = serializer_xml
repo = serializer_xml
etc...
```

This is all just an example – in your environment, you may have more complex needs – or even some weird ones.

Often folks ask about whether we can save and read from LDAP, though currently such a serializer is not implemented, though we might be interested in it if it was performant enough.

One Note of Warning

The "settings" serializer should always be "serializer_catalog", or at least should read `/var/lib/cobbler/settings` and treat it as a YAML file. Don't change it unless you know what you are doing, as that file (in YAML format) is packaged as part of the Cobbler RPM.

Future versions of Cobbler may change this default, and revert to using the YAML config only if no JSON config is found.

Notes on serializer_catalog

Serializer catalog will save individual files in:

```
/var/lib/cobbler/config/distros.d
/var/lib/cobbler/config/profiles.d
/var/lib/cobbler/config/systems.d
etc...
```

Files are named after the name of each object, for instance:

```
/var/lib/cobbler/config/systems.d/foo.json
```

On EL 4 and before, the simplejson implementation has some unicode issues, so YAML is still the default on those systems. YAML is significantly slower, so this is more reason to install Cobbler on EL 5 and later. (Or rather, json is 300x faster!)

The filenames for YAML files do not have an extension.

```
/var/lib/cobbler/config/systems.d/foo
```

Cobbler knows how to upgrade YAML files to JSON if it is running on a platform that can use JSON, and will do so transparently.

4.12.2 CouchDB

Warning: This feature has been deprecated and will not be available in Cobbler 3.0.

Cobbler 2.0.x introduced support for CouchDB as alternate storage backend, primarily as a proof of concept for NoSQL style databases. Currently, support for this backend is ALPHA-quality as it has not received significant testing.

Currently, CouchDB must be configured and running on the same system as the cobblerd daemon in order for Cobbler to connect to it successfully. Additional SELinux rules may be required for this connection if SELinux is set to enforcing mode.

Serializer Setup

Add or modify the following section in the `/etc/cobbler/modules.conf` configuration file:

```
[serializers]
settings = serializer_catalog
distro = serializer_couchdb
profile = serializer_couchdb
system = serializer_couchdb
repo = serializer_couchdb
etc...
```

Note: Be sure to leave the settings serializer set to `serializer_catalog`.

4.12.3 MongoDB

Warning: This feature has been deprecated and will not be available in Cobbler 3.0.

Cobbler 2.2.x introduced support for MongoDB as alternate storage backend, due to the native use of JSON. Currently, support for this backend is BETA-quality, and it should not be used for critical production systems.

Serializer Setup

Add or modify the following section in the `/etc/cobbler/modules.conf` configuration file:

```
[serializers]
settings = serializer_catalog
distro = serializer_mongodb
```

(continues on next page)

(continued from previous page)

```
profile = serializer_mongodb
system = serializer_mongodb
repo = serializer_mongodb
etc...
```

Note: Be sure to leave the settings serializer set to serializer_catalog.

MongoDB Configuration File

The configuration file for the MongoDB serializer is `/etc/cobbler/mongodb.conf`. This is an INI-style configuration file, which has the following default entries:

```
[connection]
host = localhost
port = 27017
```

4.12.4 MySQL

Warning: This feature has been deprecated and will not be available in Cobbler 3.0.

Cobbler 2.4.0 introduced support for MySQL as alternate storage backend. Currently, support for this backend is ALPHA-quality, and it should not be used for critical production systems.

Serializer Setup

Add or modify the following section in the `/etc/cobbler/modules.conf` configuration file:

```
[serializers]
settings = serializer_catalog
distro = serializer_mysql
profile = serializer_mysql
system = serializer_mysql
repo = serializer_mysql
etc...
```

Note: Be sure to leave the settings serializer set to serializer_catalog.

MySQL Schema

The schema for the cobbler database is very simple, and essentially uses MySQL as a key/value store with a TEXT field storing the JSON for each object. The schema is as follows:

```
CREATE DATABASE cobbler;
GRANT ALL PRIVILEGES ON cobbler.* TO 'cobbler'@'%' IDENTIFIED BY 'testing123';
CREATE TABLE distro (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE profile (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE system (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE image (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE repo (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE mgmtclass (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE file (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
CREATE TABLE package (name VARCHAR(100) NOT NULL PRIMARY KEY, data TEXT)␣
↪ENGINE=innodb;
```

MySQL Configuration File

This serializer does not yet have a configuration file, and unfortunately still hard-codes certain database values in the `cobbler/modules/serializer_mysql.py` file. If you modify the privileges or database name in the schema above, you must edit the .py module as well (be sure to remove the .pyo/.pyc files for that modules) and restart cobblerd.

4.13 Using gPXE

Support for gPXE (and by extension, iPXE) was added in 2.2.3 for booting ESXi5 installations, however it can be used for other installations as well.

Note: gPXE/iPXE support is still somewhat experimental, so use it only when required.

4.13.1 Setup

First, install the gpxe/ipxe boot images package for your OS (for RHEL variants and Fedora, the package name is gpxe-bootimgs or ipxe-bootimgs). Cobbler sync does not currently copy the undionly.kpxe file to the TFTP root directory. Again, for RHEL/Fedora you'd do something like this:

```
$ cp /usr/share/ipxe/undionly.kpxe /var/lib/tftpboot/
```

4.13.2 Configuring Systems/Profiles

Via the CLI or Web UI, just enable the gpxe option. For example:

```
$ cobbler system edit --name=foo --enable-gpxe=1 --interface=eth0 --static=1
```

When using `manage_dhcp`, a new entry for systems with static interfaces will be created as follows (following a `cobbler sync`):

```
host generic1 {
    hardware ethernet xx:xx:xx:xx:xx:xx;
    if exists user-class and option user-class = "gPXE" {
        filename "http://<cobbler server ip>/cblr/svc/op/gpxe/system/foo";
    } else {
        filename "undionly.kpxe";
    }
    next-server <next-server setting>;
}
```

Profiles and systems that use DHCP for addresses are handled by the default network block included in the `dhcp.template`.

If you're not using DHCP locally, you can just use the URL above with your custom gPXE/iPXE script: `http://<cobbler server ip>/cblr/svc/op/gpxe/system/foo`.

Here is a sample of the rendered configuration:

```
#!gpxe
kernel http://<cobbler server ip>/cobbler/images/centos6.3-x86_64/vmlinuz
imgargs vmlinuz  ksdevice=bootif lang= text kssendmac  ks=http://<cobbler␣
↪server ip>/cblr/svc/op/ks/system/foo
initrd http://192.168.122.1/cobbler/images/centos6.3-x86_64/initrd.img
boot
```

4.13.3 Templates

The templates for gPXE are stored with the other PXE templates for cobbler, in `/etc/cobbler/pxe`. The default for systems/profiles is `/etc/cobbler/pxe/gpxe_system_linux.template`, which you can see is the source for the above output:

```
$ cat /etc/cobbler/pxe/gpxe_system_linux.template
#!gpxe
kernel http://$server/cobbler/images/$distro/$kernel_name
imgargs $kernel_name $append_line
initrd http://$server/cobbler/images/$distro/$initrd_name
boot
```

As with all PXE templates, the `$append_line` variable is generated internally by cobbler, and contains the kopts arguments as well as other distro defaults that may be configured.

4.14 Data revision control

Coming soon...

This section of the manual covers the more advanced use cases for Cobbler, including methods for customizing and extending cobbler without needing to write code.

4.15 PXE behaviour and tailoring

4.15.1 Menus

Cobbler will automatically generate PXE menus for all profiles it has defined. Running `cobbler sync` is required to generate and update these menus.

To access the menus, type "menu" at the "boot:" prompt while a system is PXE booting. If nothing is typed, the network boot will default to a local boot. If "menu" is typed, the user can then choose and provision any cobbler profile the system knows about.

If the association between a system (MAC address) and a profile is already known, it may be more useful to just use "system add" commands and declare that relationship in cobbler; however many use cases will prefer having a PXE system, especially when provisioning is done at the same time as installing new physical machines.

If this behavior is not desired, run `cobbler system add --name=default --profile=plugh` to default all PXE booting machines to get a new copy of the profile "plugh". To go back to the menu system, run `cobbler system remove --name=default` and then "cobbler sync" to regenerate the menus.

When using PXE menu deployment exclusively, it is not neccessary to make cobbler system records, although the two can easily be mixed.

Additionally, note that all files generated for the pxe menu configurations are templatable, so if you wish to change the color scheme or equivalent, see the files in `/etc/cobbler`.

4.15.2 Default boot behavior

If cobbler has no record of the system being booted, cobbler will configure PXE to boot to the contents of `/etc/cobbler/default.pxe` which, if unmodified, will just fall through to the local boot process.

The recommended way to specify a different default cobbler profile to PXE boot is to create an explicit system named "default". This will cause `/etc/cobbler/default.pxe` to be ignored.

```
cobbler system add --name=default --profile=boot_this
```

To restore the previous behavior remove that explicit "default" system:

```
cobbler system remove --name=default
```

It is also possible to control the default behavior for a specific network:

```
cobbler system add --name=network1 --ip-address=192.168.0.0/24 --profile=boot_
↪this
```

4.15.3 Preventing boot loops

If your machines are set to PXE first in the boot order (ahead of hard drives), change the `pxe_just_once` flag in `/etc/cobbler/settings` to 1. This will set the machines _not_ to PXE-boot on successive boots once they complete one install. To re-enable PXE for a specific system, run the following command:

```
cobbler system edit --name=name --netboot-enabled=1
```

Service Discovery (Avahi)

If the `avahi-tools` package is installed, `cobblerd` will broadcast it's presence on the network, allowing it to be discovered by koan with the `koan --server=DISCOVER` parameter.

Repo Management

This has already been covered a good bit in the command reference section.

Yum repository management is an optional feature, and is not required to provision through cobbler. However, if cobbler is configured to mirror certain repositories, it can then be used to associate profiles with those repositories. Systems installed under those profiles will then be autoconfigured to use these repository mirrors in `/etc/yum.repos.d`, and if supported (Fedora Core 6 and later) these repositories can be leveraged even within Anaconda. This can be useful if (A) you have a large install base, (B) you want fast installation and upgrades for your systems, or (C) have some extra software not in a standard repository but want provisioned systems to know about that repository.

Make sure there is plenty of space in cobbler's webdir, which defaults to `/var/www/cobbler`.

```
cobbler reposync [--tries=N] [--no-fail]
```

Cobbler reposync is the command to use to update repos as configured with `cobbler repo add`. Mirroring can take a long time, and usage of cobbler reposync prior to usage is needed to ensure provisioned systems have the files they need to actually use the mirrored repositories. If you just add repos and never run `cobbler reposync`, the repos will never be mirrored. This is probably a command you would want to put on a crontab, though the frequency of that crontab and where the output goes is left up to the systems administrator.

For those familiar with yum's reposync, cobbler's reposync is (in most uses) a wrapper around the yum command. Please use `cobbler reposync` to update cobbler mirrors, as yum's reposync does not perform all required steps. Also cobbler adds support for rsync and SSH locations, where as yum's reposync only supports what yum supports (http/ftp).

If you ever want to update a certain repository you can run:

```
cobbler reposync --only="reponame1" ...
```

When updating repos by name, a repo will be updated even if it is set to be not updated during a regular reposync operation (ex: `cobbler repo edit --name=reponame1 --keep-updated=0`).

Note that if a cobbler import provides enough information to use the boot server as a yum mirror for core packages, cobbler can set up kickstarts to use the cobbler server as a mirror instead of the outside world. If this feature is desirable, it can be turned on by setting yum_post_install_mirror to 1 in `/etc/cobbler/settings` (and running "cobbler sync"). You should not use this feature if machines are provisioned on a different VLAN/network than production, or if you are provisioning laptops that will want to acquire updates on multiple networks.

The flags `--tries=N` (for example, `--tries=3`) and `--no-fail` should likely be used when putting reposync on a crontab. They ensure network glitches in one repo can be retried and also that a failure to synchronize one repo does not stop other repositories from being synchronized.

4.16 Kickstart Tracking

Cobbler knows how to keep track of the status of kickstarting machines.

```
cobbler status
```

Using the status command will show when cobbler thinks a machine started kickstarting and when it finished, provided the proper snippets are found in the kickstart template. This is a good way to track machines that may have gone interactive (or stalled/crashed) during kickstarts.

4.17 Boot CD

Cobbler can build all of its profiles into a bootable CD image using the `cobbler buildiso` command. This allows for PXE-menu like bringup of bare metal in evnvironments where PXE is not possible. Another more advanced method is described in the koan manpage, though this method is easier and sufficient for most applications.

CHAPTER 5

Web Interface

5.1 Security Overview

This section provides an overview of Cobbler's security model for the Web UI.

5.1.1 Why Customizable Security?

See also *Web Interface*.

When manipulating cobbler remotely, either through the Web UI or the XMLRPC interface, different classes of users want different authentication systems and different workflows. It would be wrong for Cobbler to enforce any specific workflow on someone moving to Cobbler from their current systems, as it would limit where Cobbler can be deployed. So what Cobbler does is make authentication and authorization extremely pluggable, while still shipping with some very reasonable defaults.

The center of all of this revolves around a few settings in `/etc/cobbler/modules.conf`, for example:

```
[authentication]
module = authn_configfile

[authorization]
module = authn_allowall
```

The list of choices for each option is covered in depth at the links below.

5.1.2 Authentication

The authentication setting determines what external source users are checked against to see if their passwords are valid.

See *Web Authentication*.

The default setting is to deny XMLRPC access, so all users wanting remote/web access will need to pick their authentication mode.

5.1.3 Authorization

The authorization setting determines, for a user that has already passed authentication stages, what resources they have access to in Cobbler.

See *Web Authorization*.

The default is to authorize all users that have cleared the authentication stage.

5.2 Web Authentication

Authentication controls who has access to your cobbler server. Controlling the details of what they can subsequently do is covered by a second step, *Web Authorization*.

Authentication is governed by a setting in the `[authentication]` section of `/etc/cobbler/modules.conf`, whose options are as follows:

5.2.1 Deny All (Default)

```
[authentication]
module = authn_denyall
```

This disables all external XMLRPC modifications, and also disables the Cobbler Web interface. Use this if you do not want to allow any external access and do not want to use the web interface. This is the default setting in Cobbler for new installations, forcing users to decide what sort of remote security they want to have, and is intended to make sure they think about that decision, rather than having access on by default.

5.2.2 Digest

```
[authentication]
module = authn_configfile
```

This option uses a simple digest file to hold username and password information. This is a great option if you do not have a Kerberos or LDAP server to authenticate against and just want something simple.

Be sure to change your default password for the "cobbler" user as soon as you set this up:

```
htdigest /etc/cobbler/users.digest "Cobbler" cobbler
```

You can add additional users:

```
htdigest /etc/cobbler/users.digest "Cobbler" $username
```

You can also choose to delete the "cobbler" user from the file.

Digest authentication with Apache is no longer supported due to the fact that we have moved to a session/token based authentication and form-based login scheme with the new Web UI. Unfortunately, digest authentication does not work with this method, so we now recommend using PAM or one of the other authentication schemes.

5.2.3 Defer to Apache / Kerberos

```
[authentication]
module = authn_passthru
```

This option lets Apache do the authentication and Cobbler will defer to what it decides. This is how Cobbler implements RFC 4120 (https://tools.ietf.org/html/rfc4120) support. This could be modified to use other mechanisms if so desired.

Do you want to authenticate users using Cobbler's Web UI against Kerberos? If so, this is for you.

You may also be interested in authenticating against LDAP instead – see *LDAP* – though if you have Kerberos you probably want to use Kerberos.

We assume you've already got the WebUI up and running and just want to kerberize it ... if not, see *Web Interface* first, then come back here.

Passthru

Passthru authentication has been added back to Cobbler Web as of version 2.8.0. Passthru authentication allows you to setup your webserver to perform authentication and Cobbler Web will use the value of `REMOTE_USER` to determine authorization. Using this authentication module disables the normal web form authentication which was added in Cobbler Web 2.2.0. If you prefer to use web form authentication we recommend using PAM or one of the other authentication schemes.

A common reason you might want to use Passthru authentication is to provide support for single sign on authentication like Kerberos. An example of setting this up can be found DEAD-LINK.

Bonus

These steps also work for kerberizing Cobbler XMLRPC transactions provided those URLs are the Apache proxied versions as specified in `/var/lib/cobbler/httpd.conf`

Configure the Authentication and Authorization Modes

Edit `/etc/cobbler/modules.conf`:

```
[authentication]
module = authn_passthru

[authorization]
module = authz_allowall
```

Note that you may want to change the authorization later, see *Web Authorization* for more info.

A Note About Security

The `authn_passthru` mode is only as secure as your Apache configuraton. If you make the Apache configuration permit everyone now, everyone will have access. For this reason you may want to test your Apache config on a test path like `/var/www/html/test` first, before using those controls to replace your default cobbler controls.

Configure your */etc/krb5.conf*

NOTE: This is based on my file which I created during testing. Your kerberos configuration could be rather different.

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 ticket_lifetime = 24000
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 kdc_timesync = 0

[realms]
 REDHAT.COM = {
  kdc = kdc.example.com:88
  admin_server = kerberos.example.com:749
  default_domain = example.com
 }

[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM

[kdc]
 profile = /var/kerberos/krb5kdc/kdc.conf

[pam]
 debug = false
 ticket_lifetime = 36000
```

(continues on next page)

(continued from previous page)

```
 renew_lifetime = 36000
 forwardable = true
 krb4_convert = false
```

Modify your Apache configuration file

There's a section in `/etc/httpd/conf.d/cobbler.conf` that controls access to `/var/www/cobbler/web`. We are going to modify that section. Replace that specific "Directory" section with:

(Note that for Cobbler >= 2.0, the path is actually `/cobbler_web/`)

```
LoadModule auth_kerb_module   modules/mod_auth_kerb.so

<Directory "/var/www/cobbler/web/">
  SetHandler mod_python
  PythonHandler index
  PythonDebug on

  Order deny,allow
  Deny from all
  AuthType Kerberos
  AuthName "Kerberos Login"
  KrbMethodK5Passwd On
  KrbMethodNegotiate On
  KrbVerifyKDC Off
  KrbAuthRealms EXAMPLE.COM

  <Limit GET POST>
    require user \
      gooduser1@EXAMPLE.COM \
      gooduser2@EXAMPLE.COM
    Satisfy any
  </Limit>

</Directory>
```

Note that the above example configuration can be tweaked any way you want, the idea is just that we are delegating Kerberos authentication bits to Apache, and Apache will do the hard work for us.

Also note that the above information lacks KeyTab and Service Principal info for usage with the GSS API (so you don't have to type passwords in). If you want to enable that, do so following whatever kerberos documentation you like – Cobbler is just deferring to Apache for auth so you can do whatever you want. The above is just to get you started.

Restart Things And test

```
/sbin/service cobblerd restart
/sbin/service httpd restart
```

A Note About Usernames

If entering usernames and passwords into prompts, use `user@EXAMPLE.COM` not "user".

If you are using one of the authorization mechanisms that uses `/etc/cobbler/users.conf`, make sure these match and that you do not use just the short form.

Customizations

You may be interested in the Web Authorization section to further control things. For instance you can decide to let in the users above, but only allow certain users to access certain things. The authorization module can be used independent of your choice of authentication modes.

A note about restarting cobblerd

Cobblerd regenerates an internal token on restart (for security reasons), so if you restart cobblerd, you'll have to close your browser to drop the session token and then try to login again. Generally you won't be restarting cobblerd except when restarting machines and on upgrades, so this shouldn't be a problem.

5.2.4 LDAP

```
[authentication]
module = authn_ldap
```

This option authenticates against RFC 4511 (https://tools.ietf.org/html/rfc4511) using parameters from `/etc/cobbler/settings`. This is a direct connection to LDAP without relying on Apache.

By default, the Cobbler WebUI and Web services authenticate against a digest file. All users in the digest file are "in". What if you want to authenticate against an external resource? Cobbler can do that too. These instructions can be used to make it authenticate against LDAP instead.

For the purposes of these instructions, we are authenticating against a new source install of FreeIPA – though any LDAP install should work in the same manner.

Instructions

0. Install python-ldap: `yum install python-ldap`
1. In `/etc/cobbler/modules.conf` change the authn/authz sections to look like:

```
[authentication]
module = authn_ldap

[authorization]
module = authz_configfile
```

(continues on next page)

(continued from previous page)

```
The above specifies that you authenticating against LDAP and will list which␣
↪LDAP users are valid by looking at
``/etc/cobbler/users.conf``.
```

2. In `/etc/cobbler/settings`, set the following to appropriate values to configure the LDAP parts. The values below are examples that show us pointing to an LDAP server, which is not running on the cobbler box, for authentication. Note that authorization is seperate from authentication. We'll get to that later.

```
ldap_server      : "grimlock.devel.redhat.com"
ldap_base_dn     : "DC=devel,DC=redhat,DC=com"
ldap_port        : 389
ldap_tls         : 1
```

With Cobbler 1.3 and higher, you can add additional LDAP servers by separating the server names with a space in the `ldap_server` field.

3. Now we have to configure OpenLDAP to know about the cert of the LDAP server. You only have to do this once on the cobbler box, not on each client box.

```
openssl s_client -connect servername:636
```

4. Copy everything between BEGIN and END in the above output to `/etc/openldap/cacerts/ldap.pem`

5. Ensure that the CA certificate is correctly hashed

```
 cd /etc/openldap/cacerts
 ln -s ldap.pem $(openssl x509 -hash -noout -in ldap.pem).0

On Red Hat and Fedora systems this can also be done using the cacertdir\_
↪rehash command:
```

```
cacertdir_rehash /etc/openldap/cacerts
```

6. Configure `/etc/openldap/ldap.conf` to include the following:

```
TLS_CACERTDIR   /etc/openldap/cacerts
TLS_REQCERT     allow
```

7. Edit `/etc/cobbler/users.conf` to include the list of users allowed access to cobbler resources. These must match names in LDAP. The group names are just comments.

```
[dxs]
mac = ""
pete = ""
jack = ""
```

8. Done! Cobbler now authenticates against ldap instead of the digest file, and you can limit what users can edit things by changing the `/etc/cobbler/users.conf` file.

Troubleshooting LDAP

The following trick lets you test your username/password combinations outside of the web app and may prove useful in verifying that your LDAP configuration is correct. replace $VERSION with your python version, for instance 2.4 or 2.5, etc.

```
# cp /usr/lib/python$VERSION/site-packages/cobbler/demo_connect.py /tmp/demo_
↪connect.py
# python /tmp/demo_connect.py --user=username --pass=password
```

Just run the above and look at the output. You should see a traceback if problems are encountered, which may point to problems in your configuration if you specified a valid username/password. Restart cobblerd after changing `/etc/cobbler/settings` (if you're not using *Dynamic Settings*) in order for them to take effect.

5.2.5 Spacewalk

```
[authentication]
module = authn_spacewalk
```

This module allows Spacewalk to use its own specific authorization scheme to log into Cobbler, since Cobbler is a software service used by Spacewalk.

There are settings in `/etc/cobbler/settings` to configure this, for instance redhat_management_permissive if set to 1 will enable users with admin rights in Spacewalk (or RHN Satellite Server) to access Cobbler web using the same username/password combinations.

This module requires that the address of the Spacewalk/Satellite server is configured in `/etc/cobbler/settings` (`redhat_management_server`)

5.2.6 Testing

```
[authentication]
module = authn_testing
```

This is for development/debug only and should never be used in production systems. The user "testing/testing" is always let in, and no other combinations are accepted.

5.2.7 User Supplied

Copy the signature of any existing cobbler authentication *Modules* to write your own that conforms to your organization's specific security requirements. Then just reference that module name in `/etc/cobbler/modules.conf`, restart cobblerd, and you're good to go.

5.3 Web Authorization

Authorization happens after users have been *authenticated* and controls who is then allowed, or not allowed, to perform certain specific operations in cobbler.

Note that this applies to the *Web Interface* and XML-RPC (http://xmlrpc.scripting.com/) only – the local cobbler instance can be modified with the command line tool `cobbler` as the root user, regardless of authorization policy.

Authorization choices are set in the `[authorization]` section of `/etc/cobbler/modules.conf`, whose options are as follows:

5.3.1 Allow All

```
[authorization]
module = authz_allowall
```

This module permits any user who has passed through authorization successfully to do anything they need to do, regardless of username. This is a valid choice if you are authenticating against a source that only contains trusted accounts, such as the digest authentication module. But if you are authenticating against an entire company's LDAP server or Kerberos server, however, this would be a poor choice.

5.3.2 Config File

```
[authorization]
module = authz_configfile
```

This uses the simple file `/etc/cobbler/users.conf` to provide a list of what users can access the server. The format is described below in a separate section. There are no semantics given to groups and any listed user can access cobbler to make any changes requested. Users not in this file do not have access. An example use of this setting would be if you wanted to *authenticate admins against Kerberos* but kerberos contained other passwords and you wanted to allow only users present in your whitelist to be able to make changes.

5.3.3 Ownership

```
[authorization]
module = authz_ownership
```

This is similar to authz_configfile but now enforces group dynamics. Each file in the users file (format described below) belongs to a group.

The module keeps users from modifying distributions, profiles, or system records that they do not have access to. This is a good choice if you are using cobbler in a large company, have multiple levels of administrators, or want to grant access to users to control specific systems.

If a user is in a group named "admin" or "admins" they will be able to edit any object regardless of the ownership information stored on that object in Cobbler.

Here's an example of storing ownership details in Cobbler:

```
cobbler system edit --name=system-name --owner=dbagroup,pete,mac,jack
```

This policy is rather detailed, so see more at AuthorizationWithOwnership for the full details on how this works.

5.3.4 Other authorization controls

User Supplied Module

The above authentication systems aren't expected to work for everyone.

As with any CobblerModule, users can write their own, and if they wish, submit them to the mailing list for others to use. This allows for developing even finer grained access control, or adapting cobbler to more custom/unusual configurations.

Using something like `authz_ownership` as a base would probably provide a very good place to start. If you develop something interesting you think others may want to use for policy, sharing is greatly appreciated!

File Format

The file `/etc/cobbler/users.conf` is there to configure alternative authentication modes for modules like `authz_ownership` and `authz_configfile`. In the default cobbler configuration (`authz_allowall`), this file is IGNORED, as is indicated by the comments in the file.

Here's a sample file defining a few users and groups:

```
[admins]
admin = ""
cobbler = ""
mdehaan = ""

[superlab]
obiwan = ""
luke = ""

[basement]
darth = ""
```

Note that how this file is used depends entirely on what you have in `/etc/cobbler/modules.conf` (as described above in "Module Choices"). After changing this file, cobblerd must be restarted in order for the changes to take effect.

You'll note the values have the "equals quote quote" after them. These values are currently required, but ignored. Basically they are reserved for later use.

5.4 Locking down cobbler

If you want to enable the *Web Interface* for a lot of users, and don't trust all of them to know what they are doing all of the time, here are some tips on some good configuration practices to allow for configuring a server that is hard for someone to mess with in ways they shouldn't be messing with it – as defined by you and your site specific policy.

5.4.1 /etc/cobbler/modules.conf

For *Web Authentication*, choose authn_kerberos or authn_ldap if you don't have Kerberos. See *Defer to Apache / Kerberos* and *LDAP* for details on how to set those up. Failing that, using the authn_digest is perfectly fine, but don't share passwords among the users. Logging goes to `/var/log/cobbler/*.log` and can be used to see what user does what.

For [Customizable Security](Security Overview), choose authz_ownership, as this will allow users to only edit things that they create unless you declare certain users to be admins. You should then define groups for users in `/etc/cobbler/users.conf` following the format of that file, then assign objects in cobbler (like distros, etc) ownership as described in *Ownership*.

5.4.2 Firewall

For koan to work you must unblock 25150 (XMLRPC/tcp-ip) as well as HTTP 80, HTTPS 443 and TFTP 69 (tcp/udp) if you want PXE.

If you want to access read-write XMLRPC from outside the cobbler server, you'll need to unblock 25151.

Also, if applicable, unblock DHCP!

While it may be tempting to disable cobblerd, don't... cobbler uses cobblerd to generate dynamic content such as *Kickstart Templating* and this will mean nothing will work. Koan additionally communicates with cobblerd.

5.4.3 SELinux

Cobbler works with SELinux – though you should be using EL 5 or later. EL 4 is not supported since it does not have public_content_t, meaning files can't be served from Apache and TFTP at the same time.

You should install the semanage rules that `cobbler check` tells you about to ensure everything works according to plan.

Also note, you may run into some problems if you need to relocate your `/var/www_elsewhere`, which most users should not need to do.

5.4.4 Default Passwords

Run `cobbler check` and it will warn you if any of the sample kickstarts still have "cobbler" as the password. If you are using those templates, that's a problem, if not, don't worry about it, but you may want to comment out the password line to prevent them from being used.

5.4.5 Test Your Configuration

Log in as various users (create some in different groups if need be) to make sure your authorization, authentication, and/or ACLs are correct as you expect them. Then also make sure you can deploy some physical and virtual systems outside the network to ensure your firewall configurations do not cut off anything important.

5.4.6 Command Line ACLs

All of the above is about network security, if you want to run the cobbler CLI as non root, you can run `cobbler aclsetup` to grant access to non-root users, such as your friendly trusted neighborhood administrators. Be aware this grants them file access on all of cobbler's configuration. This all uses setfacl, don't chmod yourself if you can help it as the RPM takes good steps to get all of this right. Same for running setfacl yourself as there are lots of places it must be applied.

5.4.7 A Note About File Readablity and "Security"

By nature of provisioning, TFTP and HTTP and so forth are typically wide open protocols by design. This is actually a good thing due to the Catch-22 that if it was hard to install, installing would be hard. Ease of installation requires openness, so these steps above are about keeping people from changing your provisioning configurations to ways that they should not have access to change them – they are not about denying access to data in the provisioning server, such as the contents of kickstarts. If you need to be transferring sensitive files, a long "HERE" document in kickstart %post is not the place to do it. scp those later or use a config management system to move the files.

This section of the manual covers the Cobbler Web Interface. With the web user interface (WebUI), you can:

- View all of the cobbler objects and the settings
- Add and delete a system, distro, profile, or system
- Run the equivalent of a "cobbler sync"
- Edit kickstart files (which must be in `/etc/cobbler` and `/var/lib/cobbler/kickstarts`)

You cannnot (yet):

- Auto-Import media
- Do a "cobbler validateks"

The WebUI is intended to be self-explanatory and contains tips and explanations for nearly every field you can edit. It also contains links to additional documentation, including the Cobbler manpage documentation in HTML format.

5.5 Basic Setup

1. You must have installed the cobbler-web package
2. Your `/etc/cobbler/modules.conf` should look something like this:

```
[authentication]
module = authn_configfile

[authorization]
module = authz_allowall
```

3. Change the password for the 'cobbler' username:

```
htdigest /etc/cobbler/users.digest "Cobbler" cobbler
```

4. If this is not a new install, your Apache configuration for Cobbler might not be current.

```
cp /etc/httpd/conf.d/cobbler.conf.rpmnew /etc/httpd/conf.d/cobbler.conf
```

5. Now restart Apache and Cobblerd

```
/sbin/service cobblerd restart
/sbin/service httpd restart
```

6. If you use SELinux, you may also need to set the following, so that the WebUI can connect with the XMLRPC:

```
setsebool -P httpd_can_network_connect true
```

5.6 Basic setup (2.2.x and higher)

In addition to the steps above, cobbler 2.2.x has a requirement for `mod_wsgi` which, when installed via EPEL, will be disabled by default. Attempting to start httpd will result in:

```
Invalid command 'WSGIScriptAliasMatch', perhaps misspelled \
  or defined by a module not included in the server configuration
```

You can enable this module by editing `/etc/httpd/conf.d/wsgi.conf` and un-commenting the `LoadModule wsgi_module modules/mod_wsgi.so` line.

5.6.1 Next steps

It should be ready to go. From your web browser visit the URL on your bootserver that resembles: `https://bootserver.example.com/cobbler_web` and log in with the username (usually cobbler) and password that you set earlier.

Should you ever need to debug things, see the following log files:

```
/var/log/httpd/error_log
/var/log/cobbler/cobbler.log
```

5.6.2 Further setup

Cobbler authenticates all WebUI logins through `cobblerd`, which uses a configurable authentication mechanism. You may wish to adjust that for your environment. For instance, if in `modules.conf` above you choose to stay with the authn_configfile module, you may want to add your system administrator usernames to the digest file:

```
htdigest /etc/cobbler/users.digest "Cobbler" <username>
```

You may also want to refine for authorization settings.

5.6.3 Rewrite Rule for secure-http

To redirect access to the WebUI via https on an Apache webserver, you can use the following rewrite rule, probably at the end of Apache's `ssl.conf`:

```
### Force SSL only on the WebUI
<VirtualHost *:80>
    <LocationMatch "^/cobbler/web/*">
       RewriteEngine on
       RewriteRule ^(.*) https://%{SERVER_NAME}/%{REQUEST_URI} [R,L]
   </LocationMatch>
</VirtualHost>
```

CHAPTER 6

Frequently Asked Trouble Shooting Questions

This section covers some questions that frequently come up in IRC, some of which are problems, and some of which are things about Cobbler that are not really problems, but are things folks just ask questions about frequently...

See also *Frequently Asked Virtualization Trouble Shooting Questions* for virtualization specific questions.

6.1 General

6.1.1 Most Common Things To Check

Have you run Cobbler check? What did it say? Is Cobbler and koan at the most recent released stable version? Is cobblerd running? Have you tried restarting it if this is a recent upgrade or config change? If something isn't showing up, have you restarted cobblerd and run `cobbler sync` after making changes to the config file? If you can't connect or retrieve certain files, is Apache running, or have you restarted it after a new install? If there's a koan connectivity problem, are there any firewalls blocking port `25150`?

6.1.2 I am having a problem with importing repos

Trick question! one does not run `cobbler import` on repos :) Install trees contain more data than repositories. Install trees are for OS installation and are added using `cobbler import` or `cobbler distro add` if you want do something more low level. Repositories are for things like updates and additional packages. Use `cobbler repo add` to add these sources. If you accidentally import a repo URL (for instance using rsync), clean up `/var/www/cobbler/ks_mirror/name_you_used` to take care of it. Cobbler can't detect what you are importing in advance before you copy it. Thankfully `man cobbler` gives plenty of good examples for each command, `cobbler import` and `cobbler repo add` and gives example URLs and syntaxes for both.

See also Using Cobbler Import and Manage Yum Repos for further information.

6.1.3 Why do the kickstart files in /etc/cobbler look strange?

These are not actually kickstart files, they are kickstart file templates. See *Kickstart Templating* for more information.

6.1.4 How can I validate that my kickstarts are right before installing?

Try `cobbler validateks`.

6.1.5 Can I feed normal kickstart files to –kickstart ?

You can, but you need to escape any dollar signs (`$`) with (`\\$`) so the Cobbler templating engine doesn't eat them. This is not too hard, use `cobbler profile getks` and `cobbler system getks` to make sure everything renders correctly. Also `\#raw ... \#endraw` in Cheetah can be useful. More is documented on the *Kickstart Templating* page.

6.1.6 My kickstart file has problems

If it's not related to Cobbler's *Kickstart Templating* engine, and it's more of "how do I do this in pre/post", `kickstart-list` is excellent.

Redhat Mailman/Kickstart-List (https://www.redhat.com/mailman/listinfo/kickstart-list)

Otherwise, you are likely seeing a Cheetah syntax error. Learn more about Cheetah syntax at Cheetahtemplate/User Guide/Language (https://cheetahtemplate.org/users_guide/language.html) for further information.

6.1.7 I'm running into the 255 character kernel options line limit

This can be a problem. Adding a CNAME for your cobbler server that is accessible everywhere, such as "cobbler", or even "boot" can save a lot of characters over hostname.xyz.acme-corp.internal.org. It will show up twice in the kernel options line, once for the kickstart URL, and once for the kickstart URL. Save characters by not using FQDNs when possible. The IP may also be shorter in some cases. Cobbler should try to remove optional kernel args in the event of overflow (like syslog) but you still need to be careful.

(Newer kernels are supposed to not have this limit)

6.1.8 I'm getting PXE timeouts and my cobbler server is also a virtualized host and I'm using dnsmasq for DHCP

Libvirtd starts an instance of dnsmasq unrelated to the DHCP needed for cobbler to PXE – it's just there for local networking but can cause conflicts. If you want PXE to work, do not run libvirtd on the cobbler server,

or use ISC dhcpd instead. You can of course run libvirtd on any other guests in your management network, and if you don't need PXE support, running libvirtd on the cobbler server is also fine.

Alternatively you can configure your DHCP server not to listen on all interfaces: dnsmasq run by libvirtd is configured to listen on internal virbr0/192.168.122.1 only. For ISC dhcpd you can set in /etc/sysconfig/dhcpd:

```
DHCPDARGS=eth0
```

For dnsmasq you can set in `/etc/dnsmasq.conf`:

```
interface=eth0
except-interface=lo
bind-interfaces
```

6.1.9 I'm having DHCP timeouts / DHCP is slow / etc

See the Anaconda network troubleshooting page: Fedoraproject/Anaconda/Networkissues (https://fedoraproject.org/wiki/Anaconda/NetworkIssues)

This URL has "Fedora" in it, but applies equally to Red Hat and derivative distributions.

6.1.10 Cobblerd won't start

cobblerd won't start and say:

```
> Starting cobbler daemon: Traceback (most recent call last):
>   File "/usr/bin/cobblerd", line 76, in main
>     api = cobbler_api.BootAPI(is_cobblerd=True)
>   File "/usr/lib/python2.6/site-packages/cobbler/api.py", line 127, in __
↪init__
>     module_loader.load_modules()
>   File "/usr/lib/python2.6/site-packages/cobbler/module_loader.py", line 62,
↪ in load_modules
>     blip =  __import__("modules.%s" % ( modname), globals(), locals(),␣
↪[modname])
>   File "/usr/lib/python2.6/site-packages/cobbler/modules/authn_pam.py",␣
↪line 53, in <module>
>     from ctypes import CDLL, POINTER, Structure, CFUNCTYPE, cast, pointer,␣
↪sizeof
>   File "/usr/lib64/python2.6/ctypes/__init__.py", line 546, in <module>
>     CFUNCTYPE(c_int)(lambda: None)
> MemoryError
>                                                            [  OK  ]
```

Check your SELinux. Immediate fix is to disable selinux:

```
setenforce 0
```

6.2 Debugging Cobbler Web

Most of the action in cobbler happens inside cobblerd, and the web server actually talks to it over XMLRPC.

Using epdb is probably the easiest way to debug things remotely.

6.3 Hints and tips: Redhat

A collection of tips for using Cobbler to deploy and support Redhat-based machines, including CentOS, Fedora, Scientific Linux, etc.

6.3.1 Rescue Mode

Redhat-based systems offer a "rescue" mode, typically used for trying to analyse and recover after a major OS problem. The usual way of doing this is booting from a DVD and selecting "rescue" mode at the relevant point. But it is also possible to do this via Cobbler. Indeed, if the machine lacks a DVD drive, alternatives such as this are vital for attempted rescue operations.

RISK: _Because you are using this Cobbler deployment system that usually installs machines, there is the risk that this procedure could overwrite the very machine you are attempting to rescue. So it is strongly recommended that, as part of your normal workflow, you develop and periodically verify this procedure in a safe, non-production, non-emergency environment._

The example below illustrates RHEL 5.6. The detail may vary for other Redhat-like flavours.

Assumptions

- Your target machine's Cobbler network deployment is supported by exactly one active DHCP server.
- Your deployed machines are already present in Cobbler for their earlier deployment purposes.
- A deployed machine's `kopts` setting field is usually null.
- A deployed machine's `netboot-enabled` setting is false outside deployment time.

Procedure

As stated above: _verify this periodically, outside emergency times, in a non-production environment._

On the Cobbler server:

```
cobbler system edit --name=sick-machine --kopts='rescue'
cobbler system edit --name=sick-machine --netboot-enabled=true
cobbler sync
```

As always, don't forget that `cobbler sync`.

At the client "sick-machine", start a normal deployment-style network boot. During this you should eventually see:

- Usual blue screen: `Loading SCSI driver`. There may be a couple of similar screens.
- Usual blue screen: `Sending request for IP information for eth0...`. (The exact value of that "eth0" is dependent on your machine.)
- Usual blue screen: repeat `Sending request for IP...`, but this time the header bar at the top should have `Rescue Mode` appended.
- Usual back-to-black: `running anaconda` and a couple of related lines.
- Blue screen with header bar `Rescue` and options "Continue", "Read-Only", "Skip".

In particular, if the second `Sending request for IP...` screen fails to say `Rescue Mode`, it is strongly recommended that you immediately abort the process to avoid the risk of overwriting the machine.

At this point you select whichever option is appropriate for your rescue and follow the Redhat rescue procedures. (The detail is independent of, and beyond the scope of, this Cobbler procedure.)

When you have finished, on the Cobbler server nullify the rescue:

```
cobbler system edit --name=sick-machine --kopts=''
cobbler system edit --name=sick-machine --netboot-enabled=false
cobbler sync
```

6.4 Frequently Asked Virtualization Trouble Shooting Questions

This section covers some questions that frequently come up in IRC, some of which are problems, and some of which are things about Cobbler that are not really problems, but are things folks just ask questions about frequently... All related to virtualization.

See also *Frequently Asked Trouble Shooting Questions* for general items.

6.4.1 Why don't I see this Xen distribution in my PXE menu?

There are two types of installer kernel/initrd pairs. There's a normal one (for all physical installations) and a Xen paravirt one. If you `cobbler import` an install tree (say from a DVD image) and get some "xen" distributions, these distributions will then not show up in your PXE menu – just because Cobbler knows it's impossible to PXE boot them on physical hardware.

If you want to install virtual guests, read `man koan` for details and also https://koan.readthedocs.io/en/release28/installing-virtual-guests.html

If you want to install a physical host, use the standard distribution, the one without "xen" in the name. Instead, in the `%packages` section of the kickstart, add the package named `kernel-xen`.

This only applies for Xen, of course, if you are using KVM, it's simpler and there is only one installer kernel/initrd pair to worry about – the main one.

In recent versions of Fedora, the Xen kernels have merged again, so this is not a problem.

6.4.2 I'm having problems using Koan to install virtual guests

If you use virt-type xenpv, make sure the profile you are installing uses a distro with "xen" in the name. These are the paravirtualized versions of the installer kernel/initrd pair.

Make sure your host arch matches your guest arch.

If installing Xen and using virsh console or xm console, if you don't use `--nogfx` at one point the installer will appear to hang. Most likely it did not, it switched over to using VNC which you can view with virt-manager. If you would like to keep using the text console, use `--nogfx` instead. This does not apply to other virt types, only Xen.

There really aren't any KVM gotchas, other than making sure `/dev/kvm` is present (you need the right kernel module installed on the host) otherwise things will install with qemu and appear to be very slow.

See also https://koan.readthedocs.io/en/release28/installing-virtual-guests.html

6.4.3 What Is This Strange Message From Xen?

```
libvir: Xen error : Domain not found: xenUnifiedDomainLookupByUUID
libvir: Xen error : Domain not found: xenUnifiedDomainLookupByName
```

If you see the above, it's not an error. These strange messages are perfectly normal and are coming from Xen as it's looking for an existing domain. It does not come from Cobbler/koan and your installation will not be affected. We agree they are confusing but they are not coming from Cobbler or Koan.

6.4.4 VirtualBox version 4+ won't PXE boot, DHCP logs show up nothing

If you setup cobbler all correctly and you are trying to network book with PXE and you receive this error right after the VirtualBox POST:

```
FATAL: No bootable medium found! System halted.
```

Be sure to install to install the VirtualBox Extensions Pack to enable PXE boot support.

CHAPTER 7

Appendix

7.1 S390 Support

7.1.1 Introduction

Cobbler includes support for provisioning Linux on virtualized guests under z/VM, the System z hypervisor.

7.1.2 Quickstart Guide

To begin, you need to first configure a Cobbler server. Cobbler can be run on any Linux system accessible by the mainframe, including an x86 system or another System z guest. This server's primary responsibility is to host the Linux install tree(s) remotely, and maintain information about clients accessing it. For detailed instructions on configuring the Cobbler server read the Cobbler manual.

We will assume static networking is used for System z guests.

After the Cobbler server is running, and you have imported at least one s390x install tree, you can customize the default kickstart template. Cobbler provides a sample kickstart template that you can start with called `/var/lib/cobbler/kickstarts/sample.ks`. You will want to copy this sample, and modify it by adding the following snippet in `%post`:

```
$SNIPPET('post_s390_reboot')
```

Next, it's time to add a system to Cobbler. Unlike traditional PXE, where unknown clients are identified by MAC address, zPXE uses the z/VM user ID to distinguish systems. For example, to add a system with z/VM user ID ''z01":

```
cobbler system add --name z01 \
--hostname=z01.example.com --ip-address=10.10.10.100 --subnet=10.10.10.255 --
↪netmask=255.255.255.0 \
--name-servers=10.10.10.1 --name-servers-search=example.com:example2.com \
--gateway=10.10.10.254 --kopts="LAYER2=0 NETTYPE=qeth PORTNO=0 cms=None \
HOSTNAME=z01.example.com IPADDR=10.10.10.100 SUBCHANNELS=0.0.0600,0.0.0601,0.
↪0.0602 \
MTU=1500 BROADCAST=10.10.10.255 SEARCHDNS=example.com:example2.com \
NETMASK=255.255.255.0 DNS=10.10.10.1 PORTNAME=UNASSIGNED \
DASD=100-101,200 GATEWAY=10.10.10.254 NETWORK=10.10.10.0"
```

Most of the options to `cobbler system add` are self explanatory network parameters. They are fully explained in the cobbler man page (see `man cobbler`). The `--kopts` option is used to specify System z specific kernel options needed by the installer. These are the same parameters found in the PARM or CONF file of a traditional installation, and in fact will be placed into a PARM file used by zPXE. For any parameters not specified with `--kopts`, the installer will prompt you during kickstart in the 3270 console. For a truly non-interactive installation, make sure to specify at least the parameters listed above.

Now that you've added a system to Cobbler, it's time to configure zPXE, the Cobbler-specific System z PXE emulator client, which ships with Cobbler. zPXE is designed to replace PROFILE EXEC for a System z guest. Alternatively, you can simply call ZPXE EXEC from your existing PROFILE EXEC. The following example assumes the z/VM FTP server is running; however, you can also FTP from z/VM to the cobbler server. Transfer zpxe.rexx to z/VM:

```
# cd /var/lib/cobbler
# ftp zvm.example.com
==> ascii
==> put zpxe.rexx zpxe.exec
==> bye
```

Next, logon to z/VM, and backup the current PROFILE EXEC and rename ZPXE EXEC:

```
==> rename profile exec a = execback =
==> rename zpxe exec a profile = =
```

Finally, you need to create a ZPXE CONF to specify the cobbler server hostname, as well as the default disk to IPL. Use xedit to create this file. It has only two lines.

```
==> xedit zpxe conf a

00000 * * * Top of File * * *
00001 HOST example.server.com
00002 IPLDISK 100
00003 * * * End of File * * *
```

zPXE is now configured. The client will attempt to contact the server at each logon. If there is a system record available, and it is set to be reinstalled, zPXE will download the necessary files and begin the kickstart.

To schedule an install, run the following command on the cobbler server:

```
cobbler system edit --name z01 --netboot-enabled 1 --profile RHEL-5-Server-U1-
↪s390x
```

7.1.3 Internals: How It Works

Now let's take a look at how zPXE works. First, it defines a 50 MB VDISK, which is large enough to hold a kernel and initial RAMdisk, and enough free space to convert both files to 80-character width fixed record length. Since VDISK is used, zPXE does not require any writeable space on the user's 191(A) disk. This makes it possible to use zPXE as a read-only PROFILE EXEC shared among many users.

Next, the client uses the z/VM TFTP client to contact the server specified in ZPXE CONF. It attempts to retrieve, in the following order:

1. **/s390x/s_systemname, if found, the following files will be downloaded:**
    - /s390x/s_systemname_parm
    - /s390x/s_systemname_conf
2. /s390x/profile_list

When netboot is enabled on the cobbler server, it places a file called `s_systemname` (where `systemname` is a z/VM user ID) into `/var/lib/tftpboot/s390x/` which contains the following lines:

```
/images/RHEL-5-Server-U3-s390x/kernel.img
/images/RHEL-5-Server-U3-s390x/initrd.img
ks=http://cobbler.example.com/cblr/svc/op/ks/system/z01
```

The file parameter file (`s_systemname_parm`) is intended for kernel options, and may also contain network-specific information for the guest. The config file (`s_systemname_conf`) is intended for CMS specific configuration. It is currently unused, as the parm file contains everything necessary for install. However, it is maintained as a placeholder for additional functionality.

A sample parameter file looks like this:

```
LAYER2=0 NETTYPE=qeth PORTNO=0 ip=False MTU=1500
SEARCHDNS=search.example.com DNS=192.168.5.1 GATEWAY=192.168.5.254
DASD=100-101,200 NETWORK=192.168.5.0 RUNKS=1 cmdline root=/dev/ram0
HOSTNAME=server.example.com IPADDR=192.168.5.2
SUBCHANNELS=0.0.0600,0.0.0601,0.0.0602 BROADCAST=192.168.5.255
NETMASK=255.255.255.0 PORTNAME=UNASSIGNED ramdisk_size=40000 ro cms
```

NOTE: The parameter file has several restrictions on content. The most notable restrictions are listed below. For a complete list of restrictions, refer to Redhat Access (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/installation_guide/s1-s390-steps-vm).

- The parameter file should contain no more than 80 characters per line.
- The VM reader has a limit of 11 lines for the parameter file (for a total of 880 characters).

If there is no system record available on the server, or if netboot is not enabled, zPXE will attempt to retrieve the file `profile_list`, containing a list of all available install trees. These are presented in the form of a menu which is displayed at each logon. If a profile is chosen, zPXE downloads the appropriate kernel and initial RAMdisk and begins the installation. Note that since these are generic profiles, there is no network-specific information available for this guest, so you will be prompted for this information in the 3270 console during installation.

If you press Enter at the menu without choosing a profile, zPXE will IPL the default disk specified in ZPXE CONF. If the guest is XAUTOLOG'd (logged on disconnected by another user), zPXE will check for the presence of a system record. If not found, the default disk is IPL'd with no profile list shown.

7.2 Power PC support

Cobbler includes support for provisioning Linux on PowerPC systems. This document will address how cobbler PowerPC support differs from cobbler support for more common architectures, including i386 and x86_64.

7.2.1 Setup

Support for network booting PowerPC systems is much like support for network booting x86 systems using PXE. However, since PXE is not available for PowerPC, yaboot (http://yaboot.ozlabs.org) is used to network boot your PowerPC systems. To start, you must adjust the boot device order on your system so that a network device is first. On x86-based architectures, this configuration change would be accomplished by entering the BIOS. However, Open Firmware (https://en.wikipedia.org/wiki/Open_Firmware) is often used in place of a BIOS on PowerPC platforms. Different PowerPC platforms offer different methods for accessing Open Firmware. The common procedures are outlined at Open Firmware (Access) (https://en.wikipedia.org/wiki/Open_Firmware#Access). The following example demonstrates updating the boot device order.

Once at an Open Firmware prompt, to display current device aliases use the `devalias` command. For example:

```
0 > devalias
ibm,sp                /vdevice/IBM,sp@4000
disk                  /pci@800000020000002/pci@2,4/pci1069,b166@1/scsi@1/sd@5,0
network               /pci@800000020000002/pci@2/ethernet@1
net                   /pci@800000020000002/pci@2/ethernet@1
network1              /pci@800000020000002/pci@2/ethernet@1,1
scsi                  /pci@800000020000002/pci@2,4/pci1069,b166@1/scsi@0
nvram                 /vdevice/nvram@4002
rtc                   /vdevice/rtc@4001
screen                /vdevice/vty@30000000
 ok
```

To display the current boot device order, use the `printenv` command. For example:

```
0 > printenv boot-device
-------------- Partition: common -------- Signature: 0x70 ---------------
boot-device              /pci@800000020000002/pci@2,3/ide@1/disk@0 /
↪pci@800000020000002/pci@2,4/pci1069,b166@1/scsi@1/sd@5,0
 ok
```

To add the device with alias **network** as the first boot device, use the `setenv` command. For example:

```
0 > setenv boot-device network /pci@800000020000002/pci@2,3/ide@1/disk@0 /
↪pci@800000020000002/pci@2,4/pci1069,b166@1/scsi@1/sd@5,0
```

Your system is now configured to boot off of the device with alias **network** as the first boot device. Should booting off this device fail, your system will fallback to the next device listed in the **boot-device** Open Firmware settings.

7.2.2 System-based configuration

To begin, you need to first configure a Cobbler server. Cobbler can be run on any Linux system accessible by the your PowerPC system, including an x86 system, or another PowerPC system. This server's primary responsibility is to host the Linux install tree(s) remotely, and maintain information about clients accessing it. For detailed instructions on configuring the Cobbler server, see the manual.

Next, it's time to add a system to cobbler. The following command will add a system named *ibm-505-lp1* to cobbler. Note that the cobbler profile specified (*F-11-GOLD-ppc64*) must already exist.

```
cobbler system add --name ibm-505-lp1 --hostname ibm-505-lp1.example.com \
  --profile F-11-GOLD-ppc64 --kopts "console=hvc0 serial" \
  --interface 0 --mac 00:11:25:7e:28:64
```

Most of the options to cobbler system add are self explanatory network parameters. They are fully explained in the cobbler man page (see man cobbler). The `--kopts` option is used to specify any system-specific kernel options required for this system. These will vary depending on the nature of the system and connectivity. In the example above, I chose to redirect console output to a device called *hvc0* which is a specific console device available in some virtualized guest environments (including KVM and PowerPC virtual guests).

In the example above, only one MAC address was specified. If network booting from additional devices is desired, you may wish to add more MAC addresses to your system configuration in cobbler. The following commands demonstrate adding additional MAC addresses:

```
cobbler system edit --name ibm-505-lp1 --interface 1 --mac 00:11:25:7e:28:65
cobbler system edit --name ibm-505-lp1 --interface 2 --mac 00:0d:60:b9:6b:c8
```

Note: Providing a MAC address is required for proper network boot support using yaboot.

7.2.3 Profile-based configuration

Profile-based network installations using yaboot are not available at this time. OpenFirmware is only able to load a bootloader into memory once. Once, yaboot is loaded into memory from a network location, you are not able to exit and load an on-disk yaboot. Additionally, yaboot requires specific device locations in order to properly boot. At this time there is no *local* boot target as there are in PXE configuration files.

7.2.4 Troubleshooting

OpenFirmware Ping test

If available, some PowerPC systems offer a management interface available from the boot menu or accessible from OpenFirmware directly. On IBM PowerPC systems, this interface is called SMS.

To enter SMS while your IBM PowerPC system is booting, press *1* when prompted during boot up. A sample boot screen is shown below:

```
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM

          1 = SMS Menu                          5 = Default Boot List
          8 = Open Firmware Prompt              6 = Stored Boot List


     Memory      Keyboard     Network     SCSI
```

To enter SMS from an OpenFirmware prompt, type:

```
dev /packages/gui obe
```

Once you've entered the SMS, you should see an option menu similar to:

```
 SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
------------------------------------------------------------------------------
↪-
 Main Menu
 1.   Select Language
 2.   Setup Remote IPL (Initial Program Load)
 3.   Change SCSI Settings
 4.   Select Console
 5.   Select Boot Options
```

To perform the ping test:

1. Select *Setup Remote IPL*
2. Select the appropriate network device to use

```
------------------------------------------------------------------------------
↪-
 NIC Adapters
      Device                            Location Code                 Hardware
                                                                      Address
 1.  Port 1 - IBM 2 PORT 10/100/100  U789F.001.AAA0060-P1-T1  0011257e2864
 2.  Port 2 - IBM 2 PORT 10/100/100  U789F.001.AAA0060-P1-T2  0011257e2865
```

3. Select *IP Parameters*

```
------------------------------------------------------------------------------
↪-
 Network Parameters
Port 1 - IBM 2 PORT 10/100/1000 Base-TX PCI-X Adapter: U789F.001.AAA0060-P1-T1
 1.   IP Parameters
 2.   Adapter Configuration
 3.   Ping Test
 4.   Advanced Setup: BOOTP
```

4. Enter your local network settings

```
------------------------------------------------------------------------------
↪-
 IP Parameters
Port 1 - IBM 2 PORT 10/100/1000 Base-TX PCI-X Adapter: U789F.001.AAA0060-P1-T1
 1.   Client IP Address                    [0.0.0.0]
 2.   Server IP Address                    [0.0.0.0]
 3.   Gateway IP Address                   [0.0.0.0]
 4.   Subnet Mask                          [0.0.0.0]
```

5. When complete, press *Esc*, and select *Ping Test*

The results of this test will confirm whether your network settings are functioning properly.

Confirm Cobbler Settings

Is your system configured to netboot? Confirm this by using the following command:

```
# cobbler system report --name ibm-505-lp1 | grep netboot
netboot enabled?      : True
```

Confirm Cobbler Configuration Files

Cobbler stores network boot information for each MAC address associated with a system. When a PowerPC system is configured for netbooting, a cobbler will create the following two files inside the tftp root directory:

- `ppc/01-<MAC\_ADDRESS\> - symlink to the ../yaboot`
- `etc/01-<MAC\_ADDRESS\> - a yaboot.conf configuration`

Confirm that the expected boot and configuration files exist for each MAC address. A sample configuration is noted below:

```
# for MAC in $(cobbler system report --name ibm-505-lp1 | grep mac | gawk '
↪{print $4}' | tr ':' '-');
do
    ls /var/lib/tftpboot/{ppc,etc}/01-$MAC ;
done
/var/lib/tftpboot/etc/01-00-11-25-7e-28-64
/var/lib/tftpboot/ppc/01-00-11-25-7e-28-64
/var/lib/tftpboot/etc/01-00-11-25-7e-28-65
/var/lib/tftpboot/ppc/01-00-11-25-7e-28-65
/var/lib/tftpboot/etc/01-00-0d-60-b9-6b-c8
/var/lib/tftpboot/ppc/01-00-0d-60-b9-6b-c8
```

Confirm Permissions

Be sure that SELinux file context's and file permissions are correct. SELinux file context information can be reset according to your system policy by issuing the following command:

```
# restorecon -R -v /var/lib/tftpboot
```

To identify any additional permissions issues, monitor the system log file `/var/log/messages` and the SELinux audit log `/var/log/audit/audit.log` while attempting to netboot your system.

Latest yaboot?

Network boot support requires a fairly recent yaboot. The yaboot included in cobbler-1.4.x may not support booting recent Fedora derived distributions. Before reporting a bug, try updating to the latest yaboot binary. The latest yaboot binary is available from Fedora rawhide at LINK-DEAD.

7.2.5 References

- Additional OpenFirmware information available at LINK-DEAD.

7.3 Tips for RHN

If you're deploying RHEL, there are a few extra kickstart and Cobbler tricks you can employ to make provisioning a snap, all consolidated in one place...

7.3.1 Importing

Download the DVD ISO's for RHN hosted. Then use `cobbler import` to import the ISO's to get an install tree.

7.3.2 Registering To RHN

RHEL has a tool installed called rhnreg_ks that you may not be familiar with. It's what you call in the %post of a kickstart file to make a system automatically register itself with Satellite or the RHN Hosted offering.

You may want to read up on rhnreg_ks for all the options it provides, but Cobbler ships with a snippet ("redhat_register") that can help you register systems. It should be in the `/var/lib/cobbler/kickstarts/sample*.ks` files by default, for you to look at. It is configured by various settings in `/etc/cobbler/settings`.

7.3.3 Authenticating XMLRPC / Web users against Satellite / Spacewalk's API

In `/etc/cobbler/modules.conf`, if you are using authn_spacewalk for authentication, Cobbler can talk to Satellite (5.3 and later) or Spacewalk for authentication. Authentication is cleared when users have the role "org_admin", or "kickstart_admin" roles. Authorization can later be supplied via cobbler modules as normal, for example, `authz_allowall` (default) or `authn_ownership`, but should probably be left as `authz_allowall`.

See *Why Customizable Security?*

If you are using a copy of Cobbler that came bundled with Spacewalk or Satellite Server, don't change these settings, as you will break Spacewalk/Satellite's ability to converse with Cobbler.

7.3.4 Installation Numbers

See the section called "RHEL Keys" on the KickstartSnippets page. It's a useful way to store all of your install keys in cobbler and use them automatically as needed.

7.3.5 Repository Mirroring

Cobbler has limited/experimental support for mirroring RHN-channels, see the cobbler manpage for details. Basically you just specify a `cobbler repo add` with the path "rhn://channel-name". This requires a version of yum-utils 1.0.4 or later, installed on the cobbler boot server. Only the arch of the cobbler server can be mirrored. See ManageYumRepos.

If you require better mirroring support than what yum provides, please consider Red Hat Satellite Server.

7.4 Memtest

If installed, cobbler will put an entry into all of your PXE menus allowing you to run memtest on physical systems without making changes in Cobbler. This can be handy for some simple diagnostics.

Steps to get memtest to show up in your PXE menus:

```
# yum install memtest86+
# cobbler image add --name=memtest86+ --file=/path/to/memtest86+ --image-
↪type=direct
# cobbler sync
```

memtest will appear at the bottom of your menus after all of the profiles.

7.4.1 Targeted Memtesting

However, if you already have a cobbler system record for the system, you can't get the menu. No problem!

```
cobbler image add --name=foo --file=/path/to/memtest86 --image-type=direct
cobbler system edit --name=bar --mac=AA:BB:CC:DD:EE:FF --image=foo --netboot-
↪enabled=1
```

The system will boot to memtest until you put it back to it's original profile.

CAUTION: When restoring the system back from memtest, make sure you turn it's netboot flag /off/ if you have it set to PXE first in the BIOS order, unless you want to reinstall the system!

```
cobbler system edit --name=bar --profile=old_profile_name --netboot-enabled=0
```

Naturally if you **do** want to reinstall it after running memtest, just use `--netboot-enabled=1`

7.5 Anaconda Monitoring

This page details the Anaconda Monitoring service available in cobbler. As anamon is rather distribution specific, support for it is considered deprecated at this time.

7.5.1 History

Prior to Cobbler 1.6, remote monitoring of installing systems was limited to distributions that accept the the boot argument *syslog=*. While this is supported in RHEL-5 and newer Red Hat based distributions, it has several shortcomings.

Reduces available kernel command-line length

The kernel command-line has a limited amount of space, relying on *syslog=somehost.example.com* reduces available argument space. Cobbler has smarts to not add the *syslog=* parameter if no space is available. But doing so disables remote monitoring.

Only captures syslog

The *syslog=* approach will only capture syslog-style messages. Any command-specific output (`/tmp/lvmout`, `/tmp/ks-script`, `/tmp/X.config`) or installation failure (`/tmp/anacdump.txt`) information is not sent.

Unsupported on older distros

While capturing syslog information is key for remote monitoring of installations, the anaconda (https://fedoraproject.org/wiki/Anaconda) installer only supports sending syslog data for RHEL-5 and newer distributions.

7.5.2 What is Anamon?

In order to overcome the above obstacles, the *syslog=* remote monitoring has been replaced by a python service called **anamon** (Anaconda Monitor). Anamon is a python daemon (which runs inside the installer while it is installing) that connects to the cobbler server via XMLRPC and uploads a pre-determined set of files. Anamon will continue monitoring files for updates and send any new data to the cobbler server.

7.5.3 Using Anamon

To enable anamon for your Red Hat based distribution installations, edit */etc/cobbler/settings* and set:

```
anamon_enabled: 1
```

NOTE: Enabling anamon allows an xmlrpc call to send create and update log files in the anamon directory, without authentication, so enable only if you are ok with this limitation. It could be potentially used by users to flood the log files or fill up the server, which you probably don't want in an untrusted environment. However, even so, it may be good for debugging complex installs.

You will also need to update your kickstart templates to include the following snippets.

```
%pre
$SNIPPET('pre_anamon')
```

Anamon can also send `/var/log/messages` and `/var/log/boot.log` once your provisioned system has booted. To also enable post-install boot notification, you must enable the following snippet:

```
%post
$SNIPPET('post_anamon')
```

7.5.4 Where Is Information Saved?

All anamon logs are stored in a system-specific directory under `/var/log/cobbler/anamon/systemname`. For example,

```
$ ls /var/log/cobbler/anamon/vguest3
anaconda.log  boot.log  dmesg  install.log  ks.cfg  lvmout.log  messages  sys.
↪log
```

7.5.5 Older Distributions

Anamon relies on a %pre installation script that uses a python *xmlrpc* library. The installation image used by Red Hat Enterprise Linux 4 and older distributions for **http://** installs does not provide the needed python libraries. There are several ways to get around this . . .

1. Always perform a graphical or **vnc** install - installing graphically (or by vnc) forces anaconda to download the *stage2.img* that includes graphics support **and** the required python xmlrpc library.
2. Install your system over nfs - nfs installations will also use the *stage2.img* that includes python xmlrpc support
3. Install using an *updates.img* - Provide the missing xmlrpc library by building an updates.img for use during installation. To construct an *updates.img*, follow the steps below:

```
$ dd if=/dev/zero of=updates.img bs=1k count=1440
$ mke2fs updates.img
$ tmpdir=`mktemp -d`
$ mount -o loop updates.img $tmpdir
$ mkdir $tmpdir/cobbler
$ cp /usr/lib64/python2.3/xmlrpclib.* $tmpdir/cobbler
$ cp /usr/lib64/python2.3/xmllib.* $tmpdir/cobbler
$ cp /usr/lib64/python2.3/shlex.* $tmpdir/cobbler
$ cp /usr/lib64/python2.3/lib-dynload/operator.* $tmpdir/cobbler
```

(continues on next page)

(continued from previous page)

```
$ umount $tmpdir
$ rmdir $tmpdir
```

More information on building and using an *updates.img* is available from Anaconda Updates (https://fedoraproject.org/wiki/Anaconda/Updates)

7.6 System Retirement

Using DBAN with Cobbler to automate system retirement

7.6.1 Introduction

The following method details using DBAN (https://dban.org/) with Cobbler to create a PXE boot image that will securely wipe the disk of the system being retired. This could also be used if you are shipping a disk back to the manufacturer and wanted to ensure all data is "securely" wiped.

7.6.2 Steps

DBAN 2.2.6

Retrieve the extra loader parts that DBAN 2.2.6 needs:

```
cobbler get-loaders
```

Download DBAN:

```
wget -O /tmp/dban-2.2.6_i586.iso http://prdownloads.sourceforge.net/dban/dban-
↪2.2.6_i586.iso
```

Mount the ISO and copy the kernel image file and (optionally) the boot configuration file:

```
mount -o loop,ro /tmp/dban-2.2.6_i586.iso /mnt
mkdir -p /opt/cobbler/dban-2.2.6
cp -p /mnt/dban.bzi /opt/cobbler/dban-2.2.6/
cp -p /mnt/isolinux.cfg /opt/cobbler/dban-2.2.6/
chmod -x /opt/cobbler/dban-2.2.6/*
umount /mnt
```

Add the DBAN distro and profile to Cobbler. Run sync to copy the loaders into place:

```
cobbler distro add --name=DBAN-2.2.6-i586 --kernel=/opt/cobbler/dban-2.2.6/
↪dban.bzi \
  --initrd=/opt/cobbler/dban-2.2.6/dban.bzi --kopts="nuke=dwipe silent"
cobbler profile add --name=DBAN-2.2.6-i586 --distro=DBAN-2.2.6-i586
cobbler sync
```

DBAN 1.0.7

Download DBAN:

```
wget -O /tmp/dban-1.0.7_i386.iso http://prdownloads.sourceforge.net/dban/dban-
↪1.0.7_i386.iso
```

Mount the ISO and copy the floppy disk image file:

```
mount -o loop,ro /tmp/dban-1.0.7_i386.iso /mnt
cp -p /mnt/dban_1_0_7_i386.ima /tmp/
umount /mnt
```

Mount the floppy disk image file and copy the kernel image file, initial ram disk, and (optionally) the boot configuration file:

```
mount -o loop,ro /tmp/dban_1_0_7_i386.ima /mnt
mkdir -p /opt/cobbler/dban-1.0.7
cp -p /mnt/initrd.gz /opt/cobbler/dban-1.0.7/
cp -p /mnt/kernel.bzi /opt/cobbler/dban-1.0.7/
cp -p /mnt/syslinux.cfg /opt/cobbler/dban-1.0.7/
chmod -x /opt/cobbler/dban-1.0.7/*
umount /mnt
```

Add the DBAN distro and profile to Cobbler:

```
cobbler distro add --name=DBAN-1.0.7-i386 --kernel=/opt/cobbler/dban-1.0.7/
↪kernel.bzi \
  --initrd=/opt/cobbler/dban-1.0.7/initrd.gz --kopts="root=/dev/ram0 init=/rc␣
↪nuke=dwipe floppy=0,16,cmos"
cobbler profile add --name=DBAN-1.0.7-i386 --distro=DBAN-1.0.7-i386
```

7.6.3 Test

1. Add a system to be destroyed:

```
cobbler system add --name=00:15:c5:c0:05:58 --profile=DBAN-1.0.7-i386
```

2. Sync cobbler:

```
cobbler sync
```

3. Boot the system via PXE. The DBAN menu will pop up. Select the drives and hit F10 to start the wipe.

4. Remove the system from this profile so that you don't accidentally boot and wipe in the future:

```
cobbler system remove --name=00:15:c5:c0:05:58
```

7.6.4 Notes

You can setup DBAN to autowipe the system in question by supplying the kernel option of `nuke="dwipe --autonuke"`. We are not doing it in this example because people sometimes only half-read things and it would suck to find out too late that you'd wiped a system you didn't mean to.

It should go without saying that, while it might be a mildly fun prank, you shouldn't set this to be your default pxe boot menu choice. You'll most likely get fired and/or beat up by your fellow employees.

If you do set this profile, it will show up as an option in the PXE menus. If this concerns you, set up a syslinux password by editing the templates in `/etc/cobbler` to ensure no one walks up to a system and blitzes it involuntarily. An option to keep a profile out of the PXE menu is doable if enough people request it or someone wants to submit a patch...

7.7 Booting Live CD's

Live CD's can be used for a variety of tasks. They might update firmware, run diagnostics, assist with cloning systems, or just serve up a desktop environment.

7.7.1 With Cobbler

Somewhat unintuitively, LiveCD's are booted by transforming the CD ISO's to kernel+initrd files.

Take the livecd and install livecd-tools. You may need a recent Fedora to find livecd-tools. What we are about to do is convert the live image to something that is PXEable. It will produce a kernel image and a VERY large initrd which essentially contains the entire ISO. Once this is done it is PXE-bootable, but we still have to provide the right kernel arguments.

```
livecd-iso-to-pxeboot live-image.iso
```

This will produce a subdirectory in the current directory called `./tftpboot`. You need to save the initrd and the vmlinuz from this directory, and as a warning, the initrd is as big as the live image. Make sure you have space.

```
mkdir -p /srv/livecd
cp /path/to/cwd/tftpboot/vmlinuz0 /srv/livecd/vmlinuz0
cp /path/to/cwd/tftpboot/initrd.img /srv/livecd/initrd.img
cobbler distro add --name=liveF9 --kernel=/srv/livecd/vmlinuz0 --initrd=/srv/
↪livecd/initrd.img
```

Now we must add some parameters to the kernel and create a dummy profile object. Note we are passing in some extra kernel options and telling cobbler it doesn't need many of the default ones because it can save space. Be sure the `/name-goes-here.iso` part of the path matches up with the ISO you ran livecd-iso-to-pxeboot against exactly or the booting will not be successful.

```
cobbler distro edit --name=liveF9 --kopts='root=/f9live.iso␣
↪rootfstype=iso9660 rootflags=loop !text !lang !ksdevice'
cobbler profile add --name=liveF9 --distro=liveF9
```

At this point it will work as though it is a normal "profile", though it will boot the live image as opposed to an installer image.

For instance, if we wanted to deploy the live image to all machines on a specific subnet we could do it as follows:

```
cobbler system add --name=live_network --ip-address=123.45.00.00/24 --
↪profile=liveF9
```

Or of course we could just deploy it to a specific system:

```
cobbler system add --name=xyz --mac=AA:BB:CC:DD:EE:FF --profile=liveF9
```

And of course this will show up in the PXE menus automatically as well.

7.7.2 Notes

When you boot this profile it will take a relatively long time (3-5 minutes?) and you will see a lot of dots printed on the screen. This is expected behavior as it has to transfer a large amount of data.

7.7.3 Space Considerations

The Live Images are very large. Cobbler will try to hardlink them if the vmlinuz/initrd files are on the same device, but it cannot symlink because of the way TFTP (needed for PXE) requires a chroot environment. If your distro add command takes a long time, this is because of the copy, please make sure you have the extra space in your TFTP boot directory's partition (either `/var/lib/tftpboot` or `/tftpboot` depending on OS).

7.7.4 Troubleshooting

If you boot into the live environment and it does not work right, most likely the rootflags and other parameters are incorrect. Recheck them with "cobbler distro report –name=foo"

7.8 Clonezilla Integration

Since many of us need to support non-linux systems in addition to Linux systems, some facility for support of these systems is helpful - especially if your dhcp server is pointing pxe boots to your cobbler server. PXE booting a clonezilla live CD as an option under cobbler provides a unified starting point for all system installation tasks.

7.8.1 Step-by-step (as of 2.2.2)

1. Download clonezilla live image from here here: Clonezilla Download Page (https://clonezilla.org/downloads.php). I used the ubuntu based "experimental" version because the Debian based clonezilla's don't have necessary network drivers for many Dell servers.

2. Unpack the zip file to a location on the machine, and run an add the distribution

```
cobbler distro add  --name=clonezilla1-2-22-37 --arch=x86_64 --breed=other --
↪os-version=other \
--boot-files="'$img_path/filesystem.squashfs'='<path_to_your_folder>/live/
↪filesystem.squashfs'" \
--kernel=<path_to_your_folder>/live/vmlinuz --initrd=<path_to_your_folder>/
↪live/initrd.img``
```

3. Set up the kickstart kernel options needed for booting to at least:

```
nomodeset edd=on ocs_live_run=ocs-live-general ocs_live_keymap=NONE boot=live␣
↪vga=788 noswap noprompt nosplash \
ocs_live_batch=no ocs_live_extra_param ocs_lang=en_US.UTF-8 ocs_lang=None␣
↪nolocales config
fetch=tftp://<Your_TFTP_ServerIP>/images/<your_Profile_NAME>/filesystem.
↪squashfs``
```

4. You should then be able to create a profile for the clonezilla distro and then add to the kernel options, and be able to customize the startup procedure from there using the clonezilla docs.

5. Run `cobbler sync` to set up the template for the systems you need.

7.8.2 Limitations and work needed

1. I've seen the download of the squashfs skipped on more than one occaision, resulting in a kernel panic. Not sure why this happens, but trying again fixes the problem (or could just be because I'm still experimenting too much).

2. Would be nice if clonezilla UI was integrated into cobbler somewhat (i.e. it knew the IP of the server saving the images and maybe had an ssh key so it could get access without a password).

3. **Still very experimental.**

CHAPTER 8

Indices and tables

- genindex
- modindex
- search